# Joint Communication, Computation, Caching, and Control in Big Data Multi-Access Edge Computing

Anselme Ndikumana, Nguyen H. Tran, *Senior Member, IEEE*, Tai Manh Ho, Zhu Han, *Fellow, IEEE*, Walid Saad, *Fellow, IEEE*, Dusit Niyato, *Fellow, IEEE*, and Choong Seon Hong, *Senior Member, IEEE*

**Abstract**—The concept of Multi-access Edge Computing (MEC) has been recently introduced to supplement cloud computing by deploying MEC servers to the network edge so as to reduce the network delay and alleviate the load on cloud data centers. However, compared to the resourceful cloud, MEC server has limited resources. When each MEC server operates independently, it cannot handle all computational and big data demands stemming from users devices. Consequently, the MEC server cannot provide significant gains in overhead reduction of data exchange between users devices and remote cloud. Therefore, joint Computing, Caching, Communication, and Control (4C) at the edge with MEC server collaboration is needed. To address these challenges, in this paper, the problem of joint 4C in big data MEC is formulated as an optimization problem whose goal is to jointly optimize a linear combination of the bandwidth consumption and network latency. However, the formulated problem is shown to be non-convex. As a result, a proximal upper bound problem of the original formulated problem is proposed. To solve the proximal upper bound problem, the block successive upper bound minimization method is applied. Simulation results show that the proposed approach satisfies computation deadlines and minimizes bandwidth consumption and network latency.

**Index Terms**—Communication, computation, caching, distributed control, multi-access edge computing, 5G network

## 1 Introduction

### 1.1 Background and Motivations

In recent years, wireless users have become producers and consumers of contents as their devices are now embedded with various sensors [1], which help in creating and collecting various types of data from different domains such as energy, agriculture, healthcare, transport, security, and smart homes, among others. Indeed, by the year 2020, it is anticipated that 50 billion things will be connected to the Internet, which is equivalent to 6 devices per person on the planet [2]. Therefore, the devices of wireless users will be anywhere, anytime, and connected to anything [3]. With large-scale interconnection of people and things, there will be a tremendous growth of data traffic with different characteristics (unstructured, quasi-structured, and semi-structured) whose scale, distribution, diversity, and velocity fall into a big data framework that requires big data infrastructure and analytics. Since the resources (e.g., battery power, CPU cycles, memory, and I/O data rate) of edge user devices are limited, edge user devices must offload computational tasks and big data to the cloud [4]. However, for effective big data analytics of delay sensitive and context-aware applications, there is a strong need for low-latency and reliable computation. As such, reliance on a cloud can hinder the performance of big data analytics, due to the associated overhead and end-to-end delays [3], [5].

To reduce end-to-end delay and the need for extensive user-cloud communication, *Multi-access Edge Computing (MEC)* has been introduced by the European Telecommunications Standards Institute (ETSI) as a supplement to cloud computing and mobile edge computing [6]. MEC extends cloud computing capabilities by providing IT-based services and cloud computing capabilities at the networks edges. In other words, MEC pushes 4C to the edge of the network [7]. Typically, MEC servers are deployed at the Base Stations (BSs) of a wireless network (e.g., a cellular network) for executing delay sensitive and context-aware applications in close proximity to the users [8], [9], [10].

- *A. Ndikumana and C. S. Hong are with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Republic of Korea. E-mail: {anselme, cshong}@khu.ac.kr.*
- *N. H. Tran is with the School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia, and also with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Republic of Korea. E-mail: nguyen.tran@sydney.edu.au.*
- *T. Manh Ho is with the Institut National de la Recherche Scientifique (INRS), Universit du Qubec, Montral, QC H3C 3P8, Canada, and also with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Republic of Korea. E-mail: manhtai.ho@emt.inrs.ca.*
- *Z. Han is with the Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004, and also with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Republic of Korea. E-mail: zhan2@uh.edu.*
- *W. Saad is with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, and also with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Republic of Korea. E-mail: walids@vt.edu.*
- *D. Niyato is with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore 639798, and also with the Department of Computer Science and Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, Republic of Korea. E-mail: dniyato@ntu.edu.sg.*

*Manuscript received 29 Mar. 2018; revised 12 Nov. 2018; accepted 24 Mar. 2019. Date of publication 29 Mar. 2019; date of current version 5 May 2020. (Corresponding author: Choong Seon Hong.)*
*Digital Object Identifier no. 10.1109/TMC.2019.2908403*

Data and computational task offloading to nearby MEC servers can reduce data exchange between users and the remote cloud. In other words, data will be offloaded, processed, analyzed, and cached at the edge of the network, at MEC servers, i.e., at near where data is created. To achieve this, we need to have caching storage, big data platform, and analytics software in the MEC server. However, compared to the remote cloud, an MEC server has limited resources. Therefore, in order to satisfy the users' demands, MEC servers (located in the same area) need to collaborate and share resources. Furthermore, since offloading data for being processed, analyzed, and cached at the MEC server requires communication resources, rather than considering each C (Computing, Caching, Communication, or Control) independently, we need to have a joint 4C model that reduces communication delay, computational delay, and backhaul bandwidth consumption.

### 1.2 MEC Challenges for Dealing with Big Data

The most important challenges that MEC is still facing when dealing with big data and edge analytics are:

- Users offload tasks and corresponding data with varying rates. In other words, data from multiple users may reach MEC servers too rapidly with a finite or infinite flow (e.g., streaming data), and this data needs to be processed immediately (e.g., live stream computation and caching, real-time analytics) [11]. An MEC server will find it challenging to deal with such data due to its scale, diversity, and timeliness. Therefore, for fast, parallel, and distributed processing, MEC servers must have big data platform and analytics applications for splitting data volume, distributing computations to multiple computing nodes, replicating data partitions, and recovering data when needed.
- MEC server resources are limited compared to the remote cloud [12]. Therefore, when each MEC server operates independently, it cannot efficiently handle big data stemming from users devices and significantly relieve the data exchange between users devices and the remote cloud. Therefore, to reduce the delay, cooperation among MEC servers for resource sharing and optimization of the resource utilization are needed.
- The integration of MEC with a mobile network environments raises a number of challenges related to the coordination of both MEC server and mobile network services. Therefore, we need joint 4C for big data MEC is needed.

### 1.3 Contributions

In this work, we address these challenges of joint 4C for big data processing in MEC. The main contributions of this paper are summarized as follows:

- We propose a framework for joint 4C for big data MEC, where big data computation and caching functions are performed at an MEC server instead of being sent to a remote cloud. This allows the reduction of the end-to-end delay and data exchange between users and a remote cloud.
- For satisfying users demand and efficiently executing computational tasks and data caching in big data MEC, we introduce an MEC-based collaboration space or cluster, where MEC servers located in the same cluster collaborate with each other. The aim of the collaboration in MEC is to reduce the backhaul network traffic, minimize delay in coupled 4C, and maximize resource utilization.
- In order to minimize the communication delay among MEC servers and allow collaboration, inspired by the unsupervised machine learning algorithm called the Overlapping k-Means Method (OKM) [13], we propose OKM for Collaboration Space (OKM-CS), which is an application of the standard OKM algorithm in the context of an MEC scenario. The OKM-CS allows each MEC server to participate in more than one collaboration space. A collaboration space enables collaboration among MEC servers, which is based on not only distance measurements but also the available resources.
- Within each collaboration space, we formulate a collaborative optimization problem which minimizes a linear combination of bandwidth consumption and network latency, subject to the local computation capabilities of the users, computation deadlines, and MEC resource constraints. The formulated problem is shown to be non-convex, and hence, in order to solve it, we propose a proximal upper-bound problem of the original problem and apply the Block Successive Upper-bound Minimization (BSUM) method since it is considered to be a suitable framework for big-data optimization [14].

The rest of the paper is organized as follows. In Section 2, we discuss some related works, while Section 3 presents the system model. Section 4 discusses in detail our joint 4C for big data MEC, while Section 5 provides a performance evaluation. We conclude the paper in Section 6.

## 2 Literature Review

The existing, related works can be grouped into four categories: (i) big data and caching, (ii) joint caching and computation, (iii) joint caching and communication, and (iv) joint caching, computation, and communication.

*Big Data and Caching*. In [15], the authors proposed a big data framework for mobile network optimization using data from both network features and user features. Furthermore, in [16], the authors proposed centrality measures for content caching in mobile wireless networks. However, implementing the big data framework at the network edge can be challenging due to the fact that caching spaces at edge nodes are usually small, which can potentially result in a low hit ratio. To overcome this challenge, in [9], the authors highlighted the need of having cooperative caching that allows low latency content delivery. In addition to caching, in [3], the authors tried to establish connections between big data and caching in 5G wireless networks, where statistical machine learning is applied for estimating content popularity. Other machine learning approaches are surveyed in [17].

*Joint Caching and Computation (2C)*. In [18], the authors combined caching and computation at BSs for decreasing delays occurring during communication between

applications running on user devices and a remote cloud. They developed a resource management algorithm that guides the BS to jointly schedule computation offloading and data caching allocation. In [19], the idea of low-latency computations is explored using the online secretary framework, where the computational tasks are distributed between the edge networks and cloud. Furthermore, for efficient resource usage at the BS level, in [20], the authors proposed a collaborative video caching and processing scheme in which MEC servers can assist each other. They formulated the collaborative joint caching and processing problem as an optimization problem that aims to minimize the backhaul network cost, subject to cache capacity and processing capacity constraints. In [21], the authors proposed a joint mobility aware caching and small cell base station placement framework. Also, the authors discussed the differences and relationships between caching and computation offloading. Furthermore, in [22], the authors proposed prefetching caching at WiFi Access Points (APs), where caching is based on aggregate network-level statistics and network-level mobility prediction. This helps mobile users to download contents from APs rather than to get contents from the original content servers. In addition, to address the mobile flashcrowd demands, the authors in [23] proposed a proactive caching approach for prefetching and caching contents in small cells based on user mobility prediction.

*Joint Caching and Communication (2C).* In [24], in order to significantly reduce redundant data transmissions and improve content delivery, the authors highlighted the need of having efficient content caching and distribution techniques. They proposed an optimal cooperative content caching and delivery policy in which both femtocell BSs and user equipment participate in content caching. In [25], the authors studied the problem of resource allocation along with data caching in radio access networks (RANs). They proposed a collaborative framework that leverages device-to-device (D2D) communication for implementing content caching. In [26], a communication framework related to cache-enabled heterogeneous cellular networks with D2D communication was studied. In order to satisfy quality-of-service (QoS) requirements for the users, the authors formulated a joint optimization problem that aims at maximizing the system capacity in which bandwidth resource allocation was considered. The problem of joint caching and communication for drone-enabled systems is also studied in [27].

*Joint Caching, Computation, and Communication (3C).* In [28], the authors combined 3C for designing a novel information centric heterogeneous network framework that enables content caching and computing in MEC. They considered virtualized resources, where communication, computing and caching resources can be shared among all users associated with different virtual service providers. Since MEC can enhance the computational capabilities of edge nodes, in [29], the authors formulated a computation offloading decision, resource allocation and data caching framework as an optimization problem in which the total revenue of the network is considered. Furthermore, in [30], the authors proposed an energy-efficient framework that considers joint networking, caching, and computing whose goal is to meet the requirements of the next generation of green wireless networks. Moreover, for MEC applications, in [31], the authors explored the fundamental tradeoffs between caching, computing, and communication for VR/AR applications. Finally, the work in [32] proposed a joint caching and offloading mechanism that considers task uploading and executing, computation output downloading, multi-user diversity, and multi-casting.

In [3], [15], [16], [18], [28], [29], [30], and [32], the authors consider edge caching. However, edge nodes are resources-limited as compared to the cloud. Therefore, without cooperation among edge nodes, edge caching can result in a low cache hit ratio. In order to overcome this issue, in [9] and [20], the authors proposed the idea of a collaboration space for edge nodes. However, the works in [9] and [20] do not provide any rigorous framework for analyzing the formation of collaboration spaces. Furthermore, a user may request a content format (e.g., avi), which is not available in the cache storage. Instead, the cache storage may have other content formats (e.g., mpeg) of the same content which can be converted to the desired format, by using certain computations, and then transmitted to the requesting user. This process of serving cached content after computation was not considered in [21], [24], [25], [26], [27]. Finally, the works in [18], [28], [29], [30] do not take into account any user deadlines for performing computations, which can be impractical.

To this end, our proposed approach will have several key differences from these prior approaches including: (i) while many related works (e.g.,[28], [29], [30], [31], [32]) focus on 2C and 3C, in our proposed approach, we combine 4C in big data MEC in which the computation capabilities of the user devices, computation deadline, size of input data, and MEC resource constraints are considered, (ii) The proposed collaboration between MEC servers, where MEC servers are grouped in collaboration spaces via the OKM-CS algorithm, is new in MEC, and thus is not only based on distance measurements, but also based on the availability of resources, (iii) Within each collaboration space, for solving the formulated collaborative optimization problem, we apply the BSUM method, which is not yet utilized in existing MEC solutions. The BSUM method is a novel and powerful framework for big-data optimization [14]. The BSUM method allows the decomposition of the formulated optimization problem into small subproblems which can be addressed separately and computed in parallel.

## 3 System Model

As shown in Fig. 1, we consider an MEC network composed of a set $\mathcal{M}$ of MEC servers, each of which is attached to one BS. Unless stated otherwise, we use the terms MEC server and BS interchangeably.

Each MEC server collaborates with other MEC servers by sharing resources. Therefore, we group the BSs into *collaboration spaces (i.e., clusters)*. Unless stated otherwise, we use the terms collaboration space and cluster interchangeably. In order to minimize the communication delay among MEC servers, our clustering process for BSs is based on proximity (distance) measurements, where BSs that are close enough will be grouped in the same cluster. Moreover, in our collaboration space, we focus on geographic space coverage rather than geographical space partitioning. As an example, some MEC servers in the hotspot area may want to collaborate with MEC servers that are not in the hotspot. To achieve this objective, we consider an overlapping clustering method that allows one BS to belong to more than one cluster and to share resources not only based on distance measurements but also based on resource availability and utilization.

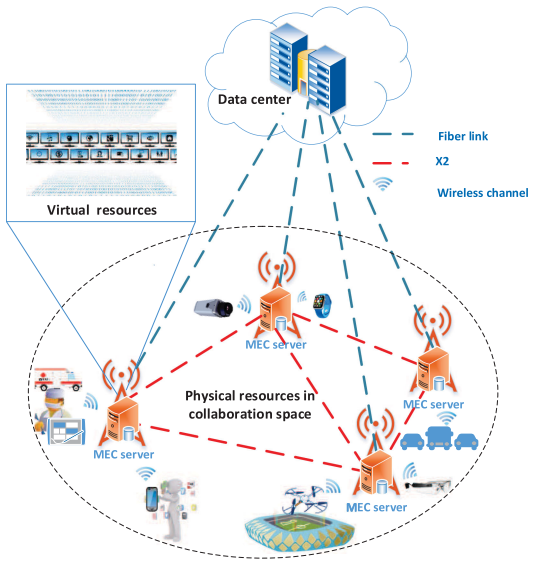


Fig. 1. Illustration of our system model.

For creating collaboration spaces, we propose OKM-CS, which is an application of the standard OKM algorithm [13] in MEC. The merit of the OKM algorithm lies in its elegant simplicity of implementation over other overlapping methods such as Weighted OKM (WOKM), Overlapping Partitioning Cluster (OPC), and Multi-Cluster Overlapping k-Means Extension (MCOKE) [33]. OKM-CS is described in Section 4, Algorithm 1.

In a collaboration space, each MEC server $m$ has both caching and computational resources that are divisible. We let $C_m$ and $P_m$ be, respectively, the cache capacity and computational capacity of MEC server $m$. In any given collaboration space, MEC servers can exchange data and tasks based on their available resources. Moreover, we assume that the MEC servers within a collaboration space belong to the same Mobile Network Operator (MNO), and this MNO has a total storage capacity $C$, and a total computation capacity $P$. The total cache storage pool for the MNO in a collaboration space is given by: $C = \sum_{m \in \mathcal{M}} C_m$, while the computation pool of the MNO is given by: $P = \sum_{m \in \mathcal{M}} P_m$.

We assume that each MEC server $m$ uses a Resource Allocation Table (RAT) for keeping track of the available resources in the collaboration space, including CPU utilization, RAM, and storage capacity. In order to facilitate joint 4C in big data MEC, in collaboration space, MEC servers exchange RAT updates. However, for the resources that are not available in a collaboration space, MEC server $m$ forwards the associated requests to the remote Data Center (DC). Therefore, for effective resource utilization, resources are sliced for being allocated to multiple users.

We consider a set $\mathcal{K}$ of users, where each user $k \in \mathcal{K}$ is connected to its nearest BS, referred to as its home BS. The set of users connected to the same BS $m \in M$ is denoted by a subset $\mathcal{K}_m \subset \mathcal{K}$. We assume that the user devices have limited resources for both computation and caching. Therefore, instead of sending resource demands to the DC, based on users demand, MEC servers can provide computation and storage resources to the users. As an example, drones in professional sports activities can cover the event scenes and send live stream videos to their nearest MEC server $m$ for live stream caching, processing, and distribution. Based on the network conditions, users demand, and device capabilities, the cached data can be served as is or after computation (e.g., video transcoding).

TABLE 1
Summary of Our Notations

| Notation | Definition |
|---|---|
| $\mathcal{M}$ | Set of MEC servers, $\lvert\mathcal{M}\rvert = M$ |
| $\mathcal{K}$ | Set of users, $\lvert\mathcal{K}\rvert = K$ |
| $C_m$ | Total cache capacity at MEC server $m \in \mathcal{M}$ |
| $P_m$ | Total computation capacity at MEC server $m \in \mathcal{M}$ |
| $s(d_k)$ | Size of data $d_k$, $\forall k \in \mathcal{K}$ |
| $\tilde{\tau}_k$ | Computation deadline, for $k \in \mathcal{K}$ |
| $\tilde{z}_k$ | Computation workload, $\forall k \in \mathcal{K}$ |
| $\lambda_m^{d_k}$ | Request arrival rate for data $d_k$ at MEC $m \in \mathcal{M}$ |
| $l_k$ | Execution latency, $\forall k \in \mathcal{K}$ |
| $E_k$ | Computation energy, $\forall k \in \mathcal{K}$ |
| $\tilde{E}_k$ | Available energy in user device $k \in \mathcal{K}$ |
| $x_k^m$ | Computation offloading decision variable, for $k \in \mathcal{K}$, and $m \in \mathcal{M}$ |
| $y_k^{m \to n}$ | Computation offloading decision variable, for $m, n \in \mathcal{M}$ |
| $w_m^k$ | Data caching decision variable, $\forall k \in \mathcal{K}_m,\ m \in \mathcal{M}$ |
| $\gamma_k^m$ | Spectrum efficiency, $\forall k \in \mathcal{K}_m$, and $m \in \mathcal{M}$ |
| $R_k^m$ | Instantaneous data rate, $\forall k \in \mathcal{K}_m$, and $m \in \mathcal{M}$ |
| $\tau_k^{k \to m}$ | Offloading delay, $\forall k \in \mathcal{K}_m$, and $m \in \mathcal{M}$ |
| $T_k$ | Task from user $k \in \mathcal{K}$ |
| $\tau_{km}^e$ | Total executing time of offloaded task, $\forall k \in \mathcal{K}_m$, $m \in \mathcal{M}$ |
| $\Theta(\boldsymbol{x}, \boldsymbol{y})$ | Total delay |
| $\Psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$ | Alleviated backhaul bandwidth |

In our model, each user device $k \in \mathcal{K}$ has an application that needs to use computation and caching resources, such as augmented reality, online gaming, crowdsensing, image processing, or CCTV video processing.

We consider a binary task offloading model in which a task is a single entity that is either computed locally at a user device or offloaded to the MEC server. For each user $k$, we define a task $T_k = (s(d_k), \tilde{\tau}_k, \tilde{z}_k),\ \forall k \in \mathcal{K}$, where $s(d_k)$ is the size of data $d_k$ from user $k$ in terms of the bits that are needed as an input of computation, $\tilde{\tau}_k$ is the task computation deadline, and $\tilde{z}_k$ is the computation workload or intensity in terms of CPU cycles per bit. Furthermore, we assume that the resource demands of different users are independent.

In order to satisfy users demand, as depicted in Fig. 2, we consider each MEC server to be a small big data infrastructure that supports the big data cloud requirements defined in [34], including $(i)$ Easy setup of virtual machines, mounting file systems, and deployment of big data platform and analytics software such as Hadoop, Spark, Storm, and Splunk; $(ii)$ Dynamic management of computation, storage, and network resources, either on physical or virtual environments; $(iii)$ Elasticity and scalability in computation, storage and network resources allocation; $(iv)$ Development, deployment and utilization of big data analytics with fast access to data and computing resources; and $(v)$ Support for multi-dimension data handling, where data may reach the MEC server in different forms and characteristics. The summary of our notations for this paper is available in Table 1.

## 4 Proposed Joint Communication, Computation, Caching, and Control

In this section, we describe, in detail, our proposed approach for joint communication, computation, caching, and distributed control in big data MEC, where MEC server resources are virtualized and shared by multiple users. Resource

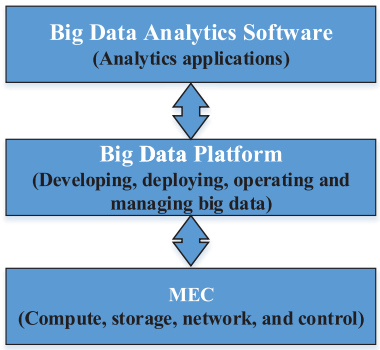


Fig. 2. Illustration of big data MEC.

demands that are not satisfied at one MEC server can be satisfied by any other MEC server in the same collaboration space.

### 4.1 Collaboration Space

For forming collaboration spaces, we propose OKM-CS. OKM-CS seeks to cluster the BSs into $r$ clusters such that the below objective function is minimized:

$$\mathcal{I}(\{\mathcal{M}_i\}_{i=1}^{r}) = \sum_{i=1}^{r} \sum_{m \in \mathcal{M}_i} \|m - \Phi(m)\|^2, \tag{1}$$

where $\mathcal{M}_i \subset \mathcal{M}$ represents the $i$th cluster. Furthermore, as defined in [13], $\Phi(m)$ defines the average of centroids ($m_{c_i}$) of the clusters to which the BS $m$ belongs, and is given by:

$$\Phi(m) = \frac{\sum_{m_{c_i} \in \mathcal{A}_i^m} m_{c_i}}{|\mathcal{A}_i^m|}, \tag{2}$$

where $\mathcal{A}_i^m$ defines multi-assignment for BS $m$: $\{m_{c_i} | m \in \mathcal{M}_i\}$, which means that $\mathcal{A}_i^m$ is a set of all centroids $m_{c_i}$ for which $m \in \mathcal{M}_i$. In OKM-CS, a centroid BS refers to a BS which is at the center of each cluster, i.e., the centroid BS is unique in each cluster. Furthermore, each BS $m$ belongs to at least one cluster, where $\bigcup_{i=1}^{r} \mathcal{M}_i = \mathcal{M}$ represents the total coverage. Moreover, BSs are assigned to centroids based on geographical locations, while the overlapping clusters allow the MEC servers to participate in different clusters so as to cooperate and share resources.

The original OKM algorithm randomly chooses $r$ clusters. However, in OKM-CS for 4C, the number of clusters is chosen based on the network topology, which is known a priori by the MNO. Furthermore, when the MNO updates its network topology, OKM-CS needs to update the clusters and their associated centroids. However, we assume that the network topology does not change frequently. The OKM-CS is presented in Algorithm 1.

Algorithm 1 starts with an initial set of $r$ clusters and centroid $\{m_{c_i}^{(0)}\}_{i=1}^{r}$, and derives new coverage $\{\mathcal{M}_i^{(0)}\}_{i=1}^{r}$. Then, it iterates by computing new assignments and new centroids $\{m_{c_i}^{(t+1)}\}_{i=1}^{r}$ leading to the new coverage $\{\mathcal{M}_i^{(t+1)}\}_{i=1}^{r}$. The iterative process continues until the convergence criterion on $\mathcal{I}(\{\mathcal{M}_i^{(t)}\}_{i=1}^{r}) - \mathcal{I}(\{\mathcal{M}_i^{(t+1)}\}_{i=1}^{r}) < \epsilon$) is satisfied, where $\epsilon$ is a small positive number. Furthermore, since our focus is on the collaboration among the MEC servers in the same collaboration space, for brevity, hereinafter, we omit the subscript on $\mathcal{M}_i$ and analyze 4C for one collaboration space.

**Algorithm 1.** OKM for Collaboration Space (OKM-CS)

1: **Input:** $\mathcal{M}$: A set of BSs with their coordinates, $t_m$: Maximum number of iterations, $\epsilon > 0$;
2: **Output:** $\{\mathcal{M}_i^{(t+1)}\}_{i=1}^{r}$ : Final cluster coverage of BSs;
3: Choose $r$ and initial clusters with $\{m_{c_i}^{(0)}\}_{i=1}^{r}$ centroid;
4: For each BS $m$, compute the assignment $\mathcal{A}_i^{m(0)}$ by assigning bs $m$ to centroid $\{m_{c_i}^{(0)}\}_{i=1}^{r}$, and derive initial coverage $\{\mathcal{M}_i^{(0)}\}_{i=1}^{r}$, such that $\mathcal{M}_i^{(0)} = \{m | m_{c_i}^{(0)} \in \mathcal{A}_i^{m(0)}\}$;
5: Initialize $t = 0$;
6: For each cluster $\mathcal{M}_i^{(t)}$, compute the new centroid, $m_{c_i}^{(t+1)}$ by grouping $\mathcal{M}_i^{(t)}$;
7: For each BS $m$ and assignment $\mathcal{A}_i^{m(t)}$, compute new assignment $\mathcal{A}_i^{m(t+1)}$ by assigning bs $m$ to centroid $\{m_{c_i}^{(t+1)}\}_{i=1}^{r}$ and derive new coverage $\{\mathcal{M}_i^{(t+1)}\}_{i=1}^{r}$;
8: If Equation (1) does not converge or $t_m > t$ or $\mathcal{I}(\{\mathcal{M}_i^{(t)}\}_{i=1}^{r}) - \mathcal{I}(\{\mathcal{M}_i^{(t+1)}\}_{i=1}^{r}) > \epsilon$, set $t = t + 1$, restart from Step 6. Otherwise, stop and consider $\{\mathcal{M}_i^{(t+1)}\}_{i=1}^{r}$ as the final clusters.

In a collaboration space, for the MEC resources, each user $k$ must submit a task demand $T_k$ to its MEC server $m$. Then, the MNO maps the demands into the resource allocation that each user $k$ requires. Therefore, to help the users prepare their demands, the MNO advertises the resources available to them as well as the sum of the demands placed in the collaboration space. However, the MNO does not reveal the demands of the users to each other.

We use $v_{km}(c_{dk}, p_{km}, R_k^m)$ to represent the resource allocation function for each user $k$ at MEC server $m$, where $c_{dk}$ is used to denote the caching resource allocation for user data of size $s(d_k)$ (i.e., $c_{dk} = s(d_k)$), $p_{km}$ is used to denote the computational resource allocation, and $R_k^m$ is used to denote the communication resource allocation.

The MNO allocates resources based on weighted proportional allocation [35], which is practical in systems such as 4G and 5G cellular networks [36], [37]. Each user $k$ receives a fraction of the resources at the MEC server $m$ based on its demand. Furthermore, when $\tilde{\tau}_k = 0$ and $\tilde{z}_k = 0$, we consider that the user needs only communication resources for offloading data $d_k$ and caching resources for caching its data. Therefore, an MEC server caches data $d_k$, and waits for the data to be requested later, where $d_k$ can be served as is or after computation. However, when $s(d_k) \neq 0$, $\tilde{\tau}_k \neq 0$, and $\tilde{z}_k \neq 0$, the MEC server computes, caches the output data of $d_k$, and returns the computation output to user $k$.

### 4.2 Communication Model

To offload a task from a user to the MEC server, the network will incur a communication cost (bandwidth). Therefore, the communication scenarios for task offloading are shown in Fig. 3 and explained next.

*Scenario (a).* For the resources available at BS $m \in \mathcal{M}$, user $k \in \mathcal{K}$ obtains resources from its MEC server over a wireless channel. We define $x_k^m \in \{0, 1\}$ as a computation offloading decision variable, which indicates whether or not user $k$ offloads a task to its home MEC server $m$ via a wireless channel (denoted by W. channel in Fig. 3).

$$x_k^m = \begin{cases} 1, & \text{if } T_k \text{ is offloaded from user } k \text{ to BS } m, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

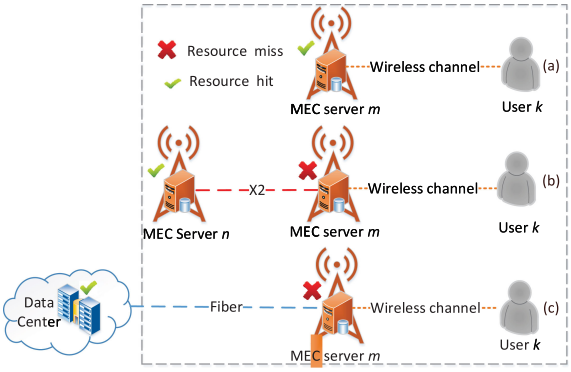


Fig. 3. Collaboration space for MEC with three typical scenarios (a), (b), and (c), which are explained in Section 4.2.

Therefore, the spectrum efficiency [28] for user device $k$ will be given by:

$$\gamma_k^m = \log_2\left(1 + \frac{\rho_k |G_k^m|^2}{\sigma_k^2}\right), \ \forall k \in \mathcal{K}, \ m \in \mathcal{M}, \tag{4}$$

where $\rho_k$ is the transmission power of user device $k$, $|G_k^m|^2$ is the channel gain between user device $k$ and BS $m$, and $\sigma_k^2$ is the power of the Gaussian noise at user $k$.

The instantaneous data rate for user device $k$ is given by:

$$R_k^m = x_k^m a_k^m B_m \gamma_k^m, \forall k \in \mathcal{K}, \ m \in \mathcal{M}, \tag{5}$$

where each user $k$ of BS $m$ is allocated a fraction $a_k^m$ ($0 \leq a_k^m \leq 1$) of bandwidth $B_m$. We assume that the spectrum of the MNO is orthogonal so that there is no interference among the users. Furthermore, we assume that user demand for offloading will only be accepted if there is enough spectrum resources to satisfy its demand.

Based on the instantaneous data rate, as defined in [38], the transmission delay for offloading a task from user $k$ to the MEC server $m$ is expressed as:

$$\tau_k^{k \to m} = \frac{x_k^m s(d_k)}{R_k^m}, \ \forall k \in \mathcal{K}_m, \tag{6}$$

where $\mathcal{K}_m$ is a set of users served by BS $m$.

*Scenario (b).* When the MEC server $m$ has insufficient resources to satisfy the user demand, after checking its RAT, BS $m$ forwards a request to another BS $n$ via an X2 link [39], in the collaboration space, which has enough resources. Therefore, users can get the resources from different MEC servers with different delay costs.

We define $y_k^{m \to n}$ as a decision variable, which indicates whether or not the task of user $k$ is offloaded from BS $m$, as follows:

$$y_k^{m \to n} = \begin{cases} 1, \text{ if } T_k \text{ of user } k \text{ is offloaded from BS } m \\ \quad \text{to a neighbor BS } n, \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

We denote by $\tau_k^{m \to n}$ the offloading delay between BS $m$ and BS $n$, which is given as follows:

$$\tau_k^{m \to n} = \frac{\sum_{k \in \mathcal{K}_m} y_k^{m \to n} s(d_k)}{\Gamma_m^n}, \ \forall m, \ n \in \mathcal{M}, \tag{8}$$

where $\Gamma_m^n$ is the X2 link capacity between BS $m$ and BS $n$.

*Scenario (c).* When the resources are not available in the whole collaboration space, BS $m$ forwards the request to the remote cloud through a wired backhaul link.

We define $y_k^{m \to DC}$ as a decision variable that indicates whether or not the task of user $k$ is offloaded by BS $m$ to the DC, as follows:

$$y_k^{m \to DC} = \begin{cases} 1, \text{ if } T_k \text{ is offloaded from BS } m \text{ to the DC}, \\ 0, \text{ otherwise.} \end{cases} \tag{9}$$

We define $\tau_k^{m \to DC}$ as the offloading delay between BS $m$ and DC $l$, where $\tau_k^{m \to DC}$ is given by:

$$\tau_k^{m \to DC} = \frac{\sum_{k \in \mathcal{K}_m} y_k^{m \to DC} s(d_k)}{\Omega_m^{DC}}, \ \forall m, \ n \in \mathcal{M}, \tag{10}$$

where $\Omega_m^{DC}$ is the link capacity between MEC server $m$ and remote DC.

## 4.3 Computation Model

### 4.3.1 Local Computation at User Device

In our model, we assume that each user device $k \in \mathcal{K}$ has a task $T_k$ that needs to use the local computation resource $P_k$ of device $k$. Therefore, the computation of task $T_k$ requires CPU energy $E_k$, where the energy consumption of CPU computation at user $k$, as defined in [38], is expressed as:

$$E_k = s(d_k) \nu \tilde{z}_k P_k^2, \ k \in \mathcal{K}, \tag{11}$$

where $\nu$ is a constant parameter that is related to the CPU hardware architecture.

In addition to the CPU energy consumption, the computation of task $T_k$ requires execution time $l_k$. Therefore, as defined in [38], the execution latency for task $T_k$ at user device $k$ is given by:

$$l_k = \frac{s(d_k)\tilde{z}_k}{P_k}. \tag{12}$$

However, when $l_k > \tilde{\tau}_k$, $\tilde{z}_k > P_k$, or $E_k > \tilde{E}_k$, where $\tilde{E}_k$ is the actual available energy at user device $k \in \mathcal{K}$, device $k$ does not have enough energy or computation resources to meet the computation deadline, and thus, user $k$ can keep the computational task until the resources become available for local computation via its device. Therefore, we define $\alpha_k \in \{0, 1\}$ as a user device status parameter for computing task $T_k$, where the value of $\alpha_k$ can be set as follows:

$$\alpha_k = \begin{cases} 0, \text{ if } \tilde{z}_k > P_k, \text{ or } l_k > \tilde{\tau}_k, \text{ or } E_k > \tilde{E}_k, \\ 1, \text{ otherwise.} \end{cases} \tag{13}$$

From the value of $\alpha_k$, the total local execution time $\tau_k^{\text{loc}}$ of task $T_k$ at user device $k$ is given by:

$$\tau_k^{\text{loc}} = \begin{cases} l_k \text{ , if } \alpha_k = 1, \text{ and } x_k^m = 0, \\ l_k + \varphi_k \text{ , if } \alpha_k = 0, \text{ and } x_k^m = 0, \\ 0, \text{ if } \alpha_k = 0, \text{ and } x_k^m = 1, \end{cases} \tag{14}$$

where $\varphi_k$ is the average waiting time of task $T_k$ until it is locally executed by device $k$.

Each user $k \in \mathcal{K}$ can compute its task $T_k$ locally on its device, when the device has enough resources, in terms of the CPU cycles, energy, or memory, whenever the user device status parameter is $\alpha_k = 1$. However, if user $k$ decides not to offload its task to an MEC server, it will experience a

computational delay $\tau_k^{\text{loc}}$. Therefore, if user $k$ cannot keep a given computational task for the future and $l_k > \tilde{\tau}_k$, $\tilde{z}_k > P_k$, $E_k > \tilde{E}_k$ ($\alpha_k = 0$), then this user $k$ can offload the task to MEC server $m$.

#### 4.3.2 Computation at MEC Server

In our model, an offloaded task $T_k$ has to be executed at the first encountered MEC server $m$ if it has enough resource. We consider $P_m$ as the available computational resources at MEC server $m \in \mathcal{M}$. Furthermore, we define $y_k^{k\to m} \in \{0,1\}$ as a decision variable, which indicates whether or not MEC server $m$ has to compute the task $T_k$ offloaded by user $k$, where $y_k^{k\to m}$ is given by:

$$y_k^{k\to m} = \begin{cases} 1, \text{ if } T_k \text{ offloaded by user } k \\ \quad \text{is computed at BS } m, \\ 0, \text{ otherwise.} \end{cases} \tag{15}$$

The computation allocation $p_{km}$ at BS $m$ can be calculated as follows:

$$p_{km} = P_m \frac{\tilde{z}_k}{\sum_{g\in\mathcal{K}_m} \tilde{z}_g}, \ \forall k \in \mathcal{K}_m, \ m \in \mathcal{M}. \tag{16}$$

At each MEC server $m$, the total computation allocations must satisfy:

$$\sum_{k\in\mathcal{K}_m} x_k^m p_{km} y_k^{k\to m} \leq P_m, \ \forall m \in \mathcal{M}. \tag{17}$$

The execution latency $l_{km}$ for task $T_k$ at MEC server $m$ is given by:

$$l_{km} = \frac{s(d_k)\tilde{z}_k}{p_{km}}. \tag{18}$$

Therefore, the total execution time for task $T_k$ that was offloaded by user $k$ at MEC server $m$ is given by:

$$\tau_{km}^e = \tau_k^{k\to m} + l_{km}, \ \forall k \in \mathcal{K}_m, \ m \in \mathcal{M}. \tag{19}$$

However, if $\tilde{z}_k > p_{km}$ or $\tau_{km}^e > \tilde{\tau}_k$ (i.e., MEC server $m$ does not have enough computational resources to meet the computation deadline), MEC server $m$ checks its RAT and offloads the task to any MEC server $n$ that has enough resources to satisfy the demand. Here, $l_{kn}$ is the execution latency for task $T_k$ at MEC server $n$ and can be calculated based on (18). Therefore, the total execution time for a task offloaded by user $k$ to MEC server $n$ becomes:

$$\tau_{kmn}^e = \tau_k^{k\to m} + \tau_k^{m\to n} + l_{kn}, \ \forall k \in \mathcal{K}_m, \text{ and } m, n \in \mathcal{M}. \tag{20}$$

When there are no available resources in a collaboration space, MEC server $m$ offloads the task to the DC. Therefore, the total execution time for task $T_k$ offloaded by user $k$ at DC becomes:

$$\tau_{kmDC}^e = \tau_k^{k\to m} + \tau_k^{m\to DC} + l_{kDC}, \ \forall k \in \mathcal{K}_m, \text{ and } m \in \mathcal{M}, \tag{21}$$

where $l_{kDC}$ can be calculated from (18). Furthermore, we find the total offloading and computation latency $\tau_k^{\text{off}}$ for task $T_k$ offloaded by user $k$ as follows:

$$\tau_k^{\text{off}} = y_k^{k\to m}\tau_{km}^e + \sum_{n\in\mathcal{M}} y_k^{m\to n}\tau_{kmn}^e + y_k^{m\to DC}\tau_{kmDC}^e, \quad \forall k \in \mathcal{K}_m, \text{ and } m \in \mathcal{M}. \tag{22}$$

In order to ensure that task $T_k$ is executed at only one location, i.e., computed locally at a user device, at one of the MEC servers, or at the remote cloud, we impose the following constraints, $\forall m \in \mathcal{M}$:

$$(1 - x_k^m) + x_k^m(y_k^{k\to m} + \sum_{n\in\mathcal{M}} y_k^{m\to n} + y_k^{m\to DC}) = 1, \tag{23}$$

$$\max\{y_k^{k\to m}, y_k^{m\to n}, y_k^{m\to DC}, \forall n\} \leq x_k^m, \ \forall k \in \mathcal{K}_m. \tag{24}$$

### 4.4 Caching Model

For an offloaded task $T_k$, MEC server $m$ caches data $d_k$. Based on the demand $\lambda_m^{d_k}$ for data $d_k$ that reaches each MEC server $m$, $d_k$ can be retrieved from the cache storage. Here, using the idea of a cacheable task defined in [40], we assume that input data of $T_k$ is cacheable. However, due to the limited cache capacity, the MNO needs to evict from the cache the least frequently reused data in order to make room for new incoming data that needs to be cached. During data replacement, the MEC server starts replacing the least frequently reused data based on the number of requests $\lambda_m^{d_k}$ satisfied, i.e., the number of cache hits. Here, the well-known Least Frequently Used (LFU) cache replacement policy [41], [42] is utilized.

We let $w_m^k \in \{0,1\}$ be the decision variable that indicates whether or not MEC server $m$ has to cache data $d_k$ of user $k$, where $w_m^k$ is given by:

$$w_m^k = \begin{cases} 1, \text{ if MEC server } m \in \mathcal{M} \text{ caches the data } d_k, \\ 0, \text{ otherwise.} \end{cases} \tag{25}$$

Here, $w_m^k$ is a cache decision policy which is essentially a rule of choosing which data $d_k$ to cache in the storage. On the other hand, LFU is a cache replacement policy. When the cache storage is full, in order to accommodate new incoming data in the cache storage, the cache replacement policy identifies the data to replace in cache storage [43]. Therefore, in LFU, the number of requests $\lambda_m^{d_k}$ satisfied by the MEC servers must be counted for identifying the least frequently reused data.

We let $C_m$ be the cache capacity available at any MEC server $m$. Therefore, the total allocation of caching resources at MEC server $m$ must satisfy:

$$\left(\sum_{k\in\mathcal{K}_m} y_k^{k\to m} + \sum_{n\neq m\in\mathcal{M}} \sum_{k\in\mathcal{K}_n} y_k^{n\to m}\right) w_m^k s(d_k) \leq C_m, \quad \forall m \in \mathcal{M}. \tag{26}$$

When MEC server $m$ does not have enough cache storage to cache data $d_k$, MEC server $m$ checks its RAT, and offloads $d_k$ to MEC server $n$ in the collaboration space (if MEC server $n$ has enough cache storage to satisfy the demand) or forwards the request to the DC. When data $d_k$ is requested at MEC server $m$, it will either be served from a cache in the collaboration space or forwarded to the DC if $d_k$ is not cached in the collaboration space.

### 4.5 Distributed Optimization Control

Next, we propose a distributed optimization control model that coordinates and integrates the communication, computation, and caching models defined in the previous sections.

In the distributed control model, we maximize the backhaul bandwidth saving (minimize the backhaul bandwidth consumption) by reducing the data exchange between MEC servers and remote DC, i.e., increasing the cache hits. Therefore, we adopt the caching reward defined in [29] as the amount of saved backhaul bandwidth given by:

$$\Psi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) = \sum_{m\in\mathcal{M}}\sum_{k\in\mathcal{K}_m} s(d_k)\lambda_m^{d_k} x_k^m (y_k^{k\rightarrow m} w_m^k + \sum_{n\in\mathcal{M}} y_k^{m\rightarrow n} w_n^k), \tag{27}$$

where the requests for data $d_k$ arrive at BS $m$ with arrival rate $\lambda_m^{d_k}$.

Here, we consider the total delay as the total amount of time that task $T_k$ takes to be completely computed (offloading delay included). For the computation cost, if user $k$ computes its task locally, then the computational delay cost of $\tau_k^{\text{loc}}$ is incurred. On the other hand, when user $k$ decides to offload the computational task to an MEC server, a total offloading and computation delay of $\tau_k^{\text{off}}$ is incurred. In order to minimize both computation delay costs ($\tau_k^{\text{loc}}$ and $\tau_k^{\text{off}}$), we formulate the total delay $\Theta(\boldsymbol{x},\boldsymbol{y})$ for the tasks computed locally at user devices, or in the MEC collaboration space, or at the remote cloud as follows:

$$\Theta(\boldsymbol{x},\boldsymbol{y}) = \sum_{m\in\mathcal{M}}\sum_{k\in\mathcal{K}_m} (1-x_k^m)\tau_k^{\text{loc}} + x_k^m \tau_k^{\text{off}}. \tag{28}$$

#### 4.5.1 Problem Formulation

We formulate the joint 4C in big data MEC as an optimization problem that jointly minimizes both bandwidth consumption and network latency as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}} \quad \Theta(\boldsymbol{x},\boldsymbol{y}) - \eta\Psi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) \tag{29}$$

$$\text{subject to:} \sum_{k\in\mathcal{K}_m} x_k^m a_k^m \le 1,\ \forall m\in\mathcal{M}, \tag{29a}$$

$$\sum_{k\in\mathcal{K}_m} x_k^m p_{km} y_k^{k\rightarrow m} \le P_m, \forall m\in\mathcal{M}, \tag{29b}$$

$$x_k^m \left(\sum_{k\in\mathcal{K}_m} y^{k\rightarrow m} + \sum_{n\neq m\in\mathcal{M}}\sum_{k\in\mathcal{K}_n} y_k^{n\rightarrow m}\right) w_m^k s(d_k) \le C_m, \tag{29c}$$

$$(1-x_k^m) + x_k^m (y_k^{k\rightarrow m} + \sum_{n\in\mathcal{M}} y_k^{m\rightarrow n} + y_k^{m\rightarrow DC}) = 1, \tag{29d}$$

$$\max\{y_k^{k\rightarrow m}, y_k^{m\rightarrow n}, y_k^{m\rightarrow DC}, \forall n\} \le x_k^m, \tag{29e}$$

where $\eta > 0$ is the weight parameter.

The constraint in (29a) guarantees that the sum of spectrum allocation to all users has to be less than or equal to the total available spectrum at each BS $m$. The constraints in (29b) and (29c) guarantee that the computation and cache resources allocated to users at each MEC server $m$ do not exceed the computation and caching resources. The constraints in (29d) and (29e) ensure that the task $T_k$ has to be executed at only one location, i.e., no duplication. Furthermore, in order to simplify the notation, we define the new objective function:

$$\mathcal{B}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) := \Theta(\boldsymbol{x},\boldsymbol{y}) - \eta\Psi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}). \tag{30}$$

The above optimization problem in (30) is difficult to solve due to its non-convex structure. Therefore, to make it convex, we use the BSUM method described in below Section 4.5.2.

#### 4.5.2 Overview of BSUM Method

BSUM is a distributed algorithm that allows parallel computing. The advantages of BSUM over centralized algorithms reside in both solution speed and problem decomposability [14]. Therefore, for introducing BSUM [44] in its standard form, we consider the following function as a block-structured optimization problem:

$$\min_{\boldsymbol{x}}\ g(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_J),\ s.t.\ \boldsymbol{x}_j\in\mathcal{Z}_j,\ \forall j\in\mathcal{J},\ j=1,\ldots,J, \tag{31}$$

where $\mathcal{Z} := \mathcal{Z}_1\times\mathcal{Z}_2\times\cdots\mathcal{Z}_J$, $g(.)$ is a continuous function, and $\mathcal{J}$ is the set of indexes. For $j=1,\ldots,J$, we consider $\mathcal{Z}_j$ as a closed convex set, and $\boldsymbol{x}_j$ as a block of variables. By applying BCD, at each iteration $t$, a single block of variables is optimized by solving the following problem:

$$\boldsymbol{x}_j^t \in \underset{\boldsymbol{x}_j\in\mathcal{Z}_j}{\operatorname{argmin}}\ g(\boldsymbol{x}_j,\ \boldsymbol{x}_{-j}^{t-1}), \tag{32}$$

where $\boldsymbol{x}_{-j}^{t-1} := (x_1^{t-1},\ldots,x_{j-1}^{t-1},x_{j+1}^{t-1},\ldots,x_j^{t-1})$, $\boldsymbol{x}_k^t = \boldsymbol{x}_k^{t-1}$ for $j\neq k$.

Both problems in (31) and (32) are difficult to solve, especially when (31) is a non-convex function, and block coordinate descent (BCD) does not always guarantee convergence. Therefore, with BSUM, at a given feasible point $\boldsymbol{y}\in\mathcal{Z}$, we can introduce the proximal upper-bound function $h(\boldsymbol{x}_j,\boldsymbol{y})$ of $g(\boldsymbol{x}_j,\boldsymbol{y}_{-j})$. The most commonly used schemes for choosing the proximal upper-bound function are quadratic upper-bound, linear upper-bound, and Jensen's upper-bound [44]. The proximal upper-bound function $h(\boldsymbol{x}_j,\boldsymbol{y})$ must satisfy following Assumption 1:

**Assumption 1.** *We make the following assumptions:*

*i)* $h(\boldsymbol{x}_j,\boldsymbol{y}) = g(\boldsymbol{y})$,
*ii)* $h(\boldsymbol{x}_j,\boldsymbol{y}) > g(\boldsymbol{x}_j,\boldsymbol{y}_{-j})$,
*iii)* $h'(\boldsymbol{x}_j,\boldsymbol{y};\boldsymbol{q}_j)|_{\boldsymbol{x}_j=\boldsymbol{y}_j} = g'(\boldsymbol{y};\boldsymbol{q}),\ \boldsymbol{y}_j+\boldsymbol{q}_j\in\mathcal{Z}_j$.

Assumptions 1(i) and 1(ii) guarantee that the proximal upper-bound function $h$ must be a global upper-bound function of the objective function $g$. Furthermore, Assumption 1 (iii) guarantees that $h(\boldsymbol{x}_j,\boldsymbol{y})$ takes steps proportional to the negative of the gradient of the objective function $g(\boldsymbol{x}_j,\boldsymbol{y}_{-j})$ in the direction $\boldsymbol{q}$, i.e., the existence of first-order derivative behavior.

For ease of presentation, we use the following proximal upper-bound, where the upper-bound is constructed by adding quadratic penalization to the objective function:

$$h(\boldsymbol{x_j},\boldsymbol{y}) = g(\boldsymbol{x}_j,\boldsymbol{y}_{-j}) + \frac{\varrho}{2}(\boldsymbol{x}_j-\boldsymbol{y}_j)^2, \tag{33}$$

where $\varrho$ is a positive penalty parameter. At each iteration $t$, the BSUM solves the proximal upper-bound function via the following update:

$$\begin{cases} \boldsymbol{x}_j^t \in \underset{\boldsymbol{x}_j\in\mathcal{Z}_j}{\operatorname{argmin}}\ h(\boldsymbol{x}_j,\boldsymbol{x}_j^{t-1}),\ \forall j\in\mathcal{J}, \\ \boldsymbol{x}_k^t = \boldsymbol{x}_k^{t-1},\ \forall k\notin\mathcal{J}. \end{cases} \tag{34}$$

There are many selection rules that can be used for selecting each coordinate $j\in\mathcal{J}$. Next, we describe the most commonly used selection rules [44]:

- BSUM for 4C with a Cyclic rule: In this selection rule, coordinates are selected in a cyclic order, i.e., $1, 2, 3, \ldots, J, 1, 2, 3, \ldots$.
- BSUM for 4C with a Gauss-Southwell rule: At each iteration $t$, the Gauss-Southwell rule selects $J$ that has a single index $j^* \in \mathcal{J}$ such that the following condition is satisfied: $j^* \in \{j \mid \|(\boldsymbol{x}_j^t - \boldsymbol{x}_j^{t-1})\| \geq q \max_{\boldsymbol{k}} \|(\boldsymbol{x}_k^t - \boldsymbol{x}_k^{t-1})\|\}$, where $j, k \in \mathcal{J}$ and $q \in [0, 1]$ is a constant.
- BSUM for 4C with a Randomized rule: At each iteration $t$, the Randomized rule defines a constant $q_{\min} \in [0, 1]$ and a probability vector $\boldsymbol{p}^t = (p_1^t \ldots p_J^t)$ that satisfies $\sum_{j \in \mathcal{J}} p_j^t = 1$ and $p_j^t \geq q_{\min}$ from which we can obtain a random index $j^* \in \mathcal{J}$ by calculating $\Pr(j \in \mathcal{J} \mid \boldsymbol{x}^{t-1}, \boldsymbol{x}^{t-2}, \ldots, \boldsymbol{x}^0 = p_j^t)$.

**Algorithm 2.** BSUM Algorithm in its Standard form [44]

1: **Input:** $\boldsymbol{x}$;
2: **Output:** $\boldsymbol{x}^*$;
3: Initialize $t = 0, \epsilon > 0$;
4: Find a feasible point $\boldsymbol{x}^0 \in \mathcal{Z}$;
5: **Repeat**;
6: Choose index set $\mathcal{J}$;
7: Let $\boldsymbol{x}_j^t \in \operatorname{argmin} h(\boldsymbol{x}_j, \boldsymbol{x}_{-j}^{t-1}),\ \forall j \in \mathcal{J}$;
8: Set $\boldsymbol{x}_k^t = \boldsymbol{x}_k^{t-1},\ \forall k \notin \mathcal{J}$;
9: $t = t + 1$;
10: **Until** $\|\frac{h_j^{(t)} - h_j^{(t+1)}}{h_j^{(t)}}\| \leq \epsilon$;
11: Then, consider $\boldsymbol{x}^* = \boldsymbol{x}_j^{(t+1)}$ as solution.

The complete structure of the BSUM algorithm is described in Algorithm 2. Algorithm 2 (BSUM) can be considered as a generalized form of BCD that optimizes block by block the upper-bound function of the original objective function. BSUM can be used for solving separable smooth or non-smooth convex optimization problems that have linear coupling constraints. To solve the family of such problems, the BSUM updates each block of variables iteratively through minimizing the proximal upper-bound function until it converges to both a coordinate-wise minimum and a stationary solution. We consider the stationary solution to be a coordinate-wise minimum, when a block of variables reaches the minimum point $\boldsymbol{x}^* = \boldsymbol{x}_j^{(t+1)}$. In other words, at stationary points, the entire vector of points cannot find a better minimum direction [44], [45], [46]. Based on [44] and [47], we can make the following remark:

**Remark 1 (Convergence).** BSUM algorithm takes $\mathcal{O}(\log(1/\epsilon))$ to converge to an $\epsilon$-optimal solution, which is sub-linear convergence.

The $\epsilon$-optimal solution $\boldsymbol{x}_j^\epsilon \in \mathcal{Z}_j$ is defined as $\boldsymbol{x}_j^\epsilon \in \{\boldsymbol{x}_j | \boldsymbol{x}_j \in \mathcal{Z}_j,\ h(\boldsymbol{x}_j, \boldsymbol{x}^t, \boldsymbol{y}^t) - h(\boldsymbol{x}_j^*, \boldsymbol{x}^t, \boldsymbol{y}^t)\} \leq \epsilon$, where $h(\boldsymbol{x}_j^*, \boldsymbol{x}^t, \boldsymbol{y}^t)$ is the optimal value of $h(\boldsymbol{x}_j, \boldsymbol{y})$ with respect to $\boldsymbol{x}_j$.

### 4.5.3 Distributed Optimization Control Algorithm

Our optimization problem in (30) is difficult to solve due to the presence of decision variables used at different locations. Therefore, we consider BSUM as a suitable candidate method for solving it in a distributed way by focusing on solving per-block subproblems. In order to apply BSUM in our distributed optimization control model, we define $\mathcal{X} \triangleq \{\boldsymbol{x} : \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}_m} x_k^m = 1, x_k^m \in [0, 1]\}$, $\mathcal{Y} \triangleq \{\boldsymbol{y} : \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}_m} y_k^{k \to m} + y_k^{m \to n} + y_k^{m \to DC} = 1, y_k^{k \to m}, y_k^{m \to n}, y_k^{m \to DC} \in [0, 1]\}$, and $\mathcal{W} \triangleq \{\boldsymbol{w} : \sum_{m \in \mathcal{M}} \sum_{k \in \mathcal{K}_m} w_m^k + w_n^k + w_{DC}^k = 1, w_m^k, w_n^k, w_{DC}^k \in [0, 1]\}$ as the feasible sets of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{w}$, respectively. In addition, in BSUM, to solve our optimization problem in (30), we need two steps:

- In the first step, we introduce a proximal function which is a convex optimization problem and an upper bound of (30) by adding quadratic penalization.
- In the second step, instead of minimizing (30) which is intractable, we minimize the proximal upper-bound function and ensure that the upper-bound function takes steps proportional to the negative of the gradient.

At each iteration $t$, $\forall j \in \mathcal{J}$, we define the proximal upper-bound function $B_j$, which is convex and the proximal upper-bound of the objective function defined in (30). In order to guarantee that the proximal upper-bound function $B_j$ is convex, we add to the objective function in (30) a quadratic penalization, as follows:

$$\mathcal{B}_j(\boldsymbol{x}_j, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{w}^{(t)}) := \mathcal{B}(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}) + \frac{\varrho_j}{2} \|(\boldsymbol{x}_j - \tilde{\boldsymbol{x}})\|^2. \tag{35}$$

(35) is the proximal upper-bound function of (30), and it can be applied to other vectors of variables $\boldsymbol{y}_j$ and $\boldsymbol{w}_j$, respectively, where $\varrho_t > 0$ is the positive penalty parameter. Furthermore, the proximal upper-bound function in (35) is a convex optimization problem due to its quadratic term $\frac{\varrho_j}{2} \|(\boldsymbol{x}_j - \tilde{\boldsymbol{x}})\|^2$. In other words, with respect to $\boldsymbol{x}_j$, $\boldsymbol{y}_j$, and $\boldsymbol{w}_j$, it has minimizers vector $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{y}}$, and $\tilde{\boldsymbol{w}}$ at each iteration $t$, which are considered to be the solution of the previous step $(t-1)$. At each iteration $t+1$, the solution is updated by solving the following optimization problems:

$$\boldsymbol{x}_j^{(t+1)} \in \min_{\boldsymbol{x}_j \in \mathcal{X}} \mathcal{B}_j(\boldsymbol{x}_j, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{w}^{(t)}), \tag{36}$$

$$\boldsymbol{y}_j^{(t+1)} \in \min_{\boldsymbol{y}_j \in \mathcal{Y}} \mathcal{B}_j(\boldsymbol{y}_j, \boldsymbol{y}^{(t)}, \boldsymbol{x}^{(t+1)}, \boldsymbol{w}^{(t)}), \tag{37}$$

$$\boldsymbol{w}_j^{(t+1)} \in \min_{\boldsymbol{w}_j \in \mathcal{W}} \mathcal{B}_j(\boldsymbol{w}_j, \boldsymbol{w}^{(t)}, \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)}). \tag{38}$$

Furthermore, (36), (37), and (38) can be solved through the use of our proposed distributed optimization control presented in Algorithm 3 for 4C, which is a modified version of the standard BSUM (Algorithm 2). For solving (36), (37), and (38), we relax the vectors of variables $\boldsymbol{x}_j$, $\boldsymbol{y}_j$, and $\boldsymbol{w}_j$ taking values in the closed interval between 0 and 1. Then, we use a threshold rounding technique described in [48] in Algorithm 3 to enforce the relaxed $\boldsymbol{x}_j$, $\boldsymbol{y}_j$, and $\boldsymbol{w}_j$ to be vectors of binary variables.

As an example, in the rounding technique, for $x_k^{m*} \in \boldsymbol{x}_j^{(t+1)}$, $x_k^{m*} \geq \theta$, where $\theta \in (0, 1)$ is a positive rounding threshold, we set $x_k^{m*}$ as follows:

$$x_k^{m*} = \begin{cases} 1, \text{ if } x_k^{m*} \geq \theta, \\ 0, \text{ otherwise.} \end{cases} \tag{39}$$

The above rounding technique can be applied to other vectors of variables $\boldsymbol{y}_j$ and $\boldsymbol{w}_j$, respectively. However, the binary solution obtained from the rounding technique may violate communication, computational, and caching resource constraints. Therefore, as described in [49], to overcome this issue after rounding, we solve the problem (35) in the form of $\mathcal{B}_j + \xi\Delta$, where constraints (29a), (29b), and (29c) are modified as follows:

$$\sum_{k\in\mathcal{K}_m} x_k^m a_k^m \leq 1+\Delta_a,\ \forall m\in\mathcal{M}, \tag{40}$$

$$\sum_{k\in\mathcal{K}_m} x_k^m p_{km} y_k^{k\rightarrow m} \leq P_m+\Delta_p, \forall m\in\mathcal{M}, \tag{41}$$

$$x_k^m\left(\sum_{k\in\mathcal{K}_m} y^{k\rightarrow m}+\sum_{n\neq m\in\mathcal{M}}\sum_{k\in\mathcal{K}_n} y_k^{n\rightarrow m}\right) w_m^k s(d_k)\leq C_m+\Delta_m, \tag{42}$$

where $\Delta_a$ is the maximum violation of communication resources constraint, $\Delta_p$ is the maximum violation of computational resources constraint, $\Delta_m$ is the maximum violation of caching resources constraint, $\Delta=\Delta_a+\Delta_p+\Delta_m$, and $\xi$ is the weight of $\Delta$. Moreover, $\Delta_a$, $\Delta_p$, and $\Delta_m$ are given by:

$$\Delta_a=\max\left\{0,\sum_{k\in\mathcal{K}_m} x_k^m a_k^m-1\right\},\ \forall m\in\mathcal{M}, \tag{43}$$

$$\Delta_p=\max\left\{0,\sum_{k\in\mathcal{K}_m} x_k^m p_{km} y_k^{k\rightarrow m}-P_m\right\},\ \forall m\in\mathcal{M}, \tag{44}$$

$$\Delta_m=\max\left\{0,x_k^m\left(\sum_{k\in\mathcal{K}_m} y^{k\rightarrow m}+\sum_{n\neq m\in\mathcal{M}}\sum_{k\in\mathcal{K}_n} y_k^{n\rightarrow m}\right) w_m^k s(d_k)-C_m\right\}. \tag{45}$$

Furthermore, if there are no violations of communication, computational, and caching resources constraints ($\Delta_a=0$, $\Delta_p=0$, and $\Delta_m=0$), the feasible solution of (35) is obtained.

Given problem $\mathcal{B}_j$ and its rounded problem $\mathcal{B}_j+\xi\Delta$, a most important measurement of the quality of rounding technique is the integrality gap which measures the ratio between the feasible solutions of $\mathcal{B}_j$ and $\mathcal{B}_j+\xi\Delta$. Therefore, based on definition and proof of integrality gap in [48], we can make the following definition:

**Definition 1 (Integrality gap).** *Given problem $\mathcal{B}_j$ (35) and its rounded problem $\mathcal{B}_j+\xi\Delta$, the integrality gap is given by:*

$$\beta=\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}}\ \frac{\mathcal{B}_j}{\mathcal{B}_j+\xi\Delta}, \tag{46}$$

*where the solution of $\mathcal{B}_j$ is obtained through relaxation of variables $\boldsymbol{x}_j$, $\boldsymbol{y}_j$, and $\boldsymbol{w}_j$, while the solution of $\mathcal{B}_j+\xi\Delta$ is obtained after rounding the relaxed variables. We consider that the best rounding is achieved, when $\beta$ ($\beta\leq 1$) is closer to 1 [48]. In other words, $\beta=1$, when $\Delta_a=0$, $\Delta_p=0$, and $\Delta_m=0$.*

In Algorithm 3 for 4C, each user device $k\in\mathcal{K}$ chooses the offloading decision $x_k^m$. If $x_k^m=1$, the user sends its demands to the nearest BS. For each demand $T_k$ received, the BS checks its RAT for its own and collaboration space resource availabilities. Algorithm 3 starts by initializing $t=0$, and setting $\epsilon$ equal to a small positive number, where $\epsilon$ is used to guarantee the $\epsilon$-optimal solution defined in [44]. Algorithm 3 then finds the initial feasible points ($\boldsymbol{x}^{(0)}$, $\boldsymbol{y}^{(0)}$, $\boldsymbol{w}^{(0)}$). Subsequently, our algorithm starts an iterative process and chooses the index set. At each iteration $t+1$, the solution is updated by solving the optimization problems (36), (37), and (38) until $\frac{\mathcal{B}_j^{(t)}-\mathcal{B}_j^{(t+1)}}{\mathcal{B}_j^{(t)}}\leq\epsilon$, i.e., it converges to an $\epsilon$-optimal solution. Algorithm 3 generates a binary solution of $\boldsymbol{x}_j^{(t+1)}$, $\boldsymbol{y}_j^{(t+1)}$, and $\boldsymbol{w}_j^{(t+1)}$ and obtains $\boldsymbol{c}$, $\boldsymbol{p}$, and $\boldsymbol{R}$ by using the rounding technique (39) and solving $\mathcal{B}_j+\xi\Delta$. Algorithm 3 also guarantees that $\mathcal{B}_j+\xi\Delta$ converges to an $\epsilon$-optimal solution. Then, after solving $\mathcal{B}_j+\xi\Delta$, Algorithm 3 calculates $\beta$, where the best rounding is achieved, when $\beta\leq 1$. Furthermore, we consider $\boldsymbol{x}^*=\boldsymbol{x}_j^{(t+1)}$, $\boldsymbol{y}^*=\boldsymbol{y}_j^{(t+1)}$, and $\boldsymbol{w}^*=\boldsymbol{w}_j^{(t+1)}$ to be stationary solution that satisfies coordinate-wise minimum. Finally, Algorithm 3 updates its RAT and sends the RAT update in its collaboration space.

The difference between the BSUM (Algorithm 2) in its standard form and the BSUM for 4C in big data MEC (Algorithm 3) resides in their implementations, where BSUM Algorithm in its standard form is based on distributed control. On the other hand, Algorithm 3 is based on both the hierarchical and distributed control models defined in [50]. In the hierarchical control model, edge devices decide on $\boldsymbol{x}$ first. Then, each MEC server $m$ acts as a controller for the users' offloaded tasks and, thus, it solves (36), (37), and (38).

In the distributed control model, each MEC server exchanges small information with other MEC servers in order to update the RAT, solve the optimization problem, and maintain the resource allocation within a tight range of available computational resources $P$ and caching resources $C$. However, in a collaboration space, there is no centralized controller that controls all MEC servers, i.e., each MEC server runs distributed optimization control algorithm (BSUM-based) for 4C. This distributed control is modeled as a dynamic feedback control model based on [51], where the RAT update at each MEC server acts as feedback with state ($\boldsymbol{x}^{(t)}$, $\boldsymbol{y}^{(t)}$, $\boldsymbol{w}^{(t)}$) at iteration $t$, which is used to determine the new state ($\boldsymbol{x}^{(t+1)}$, $\boldsymbol{y}^{(t+1)}$, $\boldsymbol{w}^{(t+1)}$) at the next iteration $t+1$. Furthermore, the optimal value ($\boldsymbol{x}_j^*$, $\boldsymbol{y}_j^*$, $\boldsymbol{w}_j^*$) is considered to be a network equilibrium or a stability point, which is the stationary solution that satisfies a coordinate-wise minimum.

**Algorithm 3.** Distributed Optimization Control Algorithm (BSUM-based) for 4C in big Data MEC

1: **Input:** $\boldsymbol{T}$: A vector of demands; $B_m$, $P_m$, and $C_m$: communication, computational and caching resources;
2: **Output:** $\boldsymbol{x}^*$, $\boldsymbol{y}^*$, $\boldsymbol{w}^*$, $\boldsymbol{c}$ : A vector of cache allocation, $\boldsymbol{p}$: A vector of computation allocation, and $\boldsymbol{R}$: A vector of communication resources allocation;
3: Each user device $k\in\mathcal{K}$ chooses the offloading decision $x_k^m$;
4: If $x_k^m=1$, user device $k\in\mathcal{K}$ sends its demand $T_k$ to BS $m\in\mathcal{M}$;
5: For each $T_k$ received at BS $m\in\mathcal{M}$, check RAT update;
6: Initialize $t=0$, $\epsilon>0$;
7: Find initial feasible points ($\boldsymbol{x}^{(0)}$, $\boldsymbol{y}^{(0)}$, $\boldsymbol{w}^{(0)}$);
8: **repeat**
9: Choose index set $\mathcal{J}$;
10: Let $\boldsymbol{x}_j^{(t+1)}\in\min_{\boldsymbol{x}_j\in\mathcal{X}}\ \mathcal{B}_j(\boldsymbol{x}_j,\boldsymbol{x}^{(t)},\boldsymbol{y}^{(t)},\boldsymbol{w}^{(t)})$;
11: Set $\boldsymbol{x}_k^{t+1}=\boldsymbol{x}_k^t, \forall k\notin\mathcal{J}$;
12: Go to Step 4, find $\boldsymbol{y}_j^{(t+1)}$, $\boldsymbol{w}_j^{(t+1)}$ by solving (37) and (38);
13: $t=t+1$;
14: **until** $\|\frac{\mathcal{B}_j^{(t)}-\mathcal{B}_j^{(t+1)}}{\mathcal{B}_j^{(t)}}\|\leq\epsilon$;
15: Generate a binary solution of $\boldsymbol{x}_j^{(t+1)}$, $\boldsymbol{y}_j^{(t+1)}$, $\boldsymbol{w}_j^{(t+1)}$ and obtain $\boldsymbol{c}$, $\boldsymbol{p}$, and $\boldsymbol{R}$ by using rounding technique (39) and solving $\mathcal{B}_j+\xi\Delta$ ;
16: Then, calculate $\beta$. If $\beta\leq 1$, consider $\boldsymbol{x}^*=\boldsymbol{x}_j^{(t+1)}$, $\boldsymbol{y}^*=\boldsymbol{y}_j^{(t+1)}$, and $\boldsymbol{w}^*=\boldsymbol{w}_j^{(t+1)}$ as a solution;
17: Update RAT, and send RAT update in collaboration space.

TABLE 2
Formation of Collaboration Spaces

| Number of BSs | $r = 100$ | $r = 500$ | $r = 1000$ | $r = 2000$ |
|---|---|---|---|---|
| Maximum | 1299 | 374 | 200 | 143 |
| Minimum | 12 | 1 | 1 | 1 |
| Average | 128 | 25 | 13 | 6 |

## 5 Simulation Results and Analysis

In this section, we present the performance evaluation of the proposed joint 4C in big data MEC, where we use Python [52] for numerical analysis.

### 5.1 Simulation Setup

For forming collaboration spaces, we use the Sitefinder dataset (BSs dataset) from Edinburgh DataShare [53]. In this dataset, we randomly select one MNO, which has 12777 BSs, through the use of the OKM-CS algorithm, we group these BSs into 1000 collaboration spaces. This clustering process requires 385.075 seconds. The choice of $r$ depends on the size of the dataset and, for our chosen dataset, we find that $r = 1000$ is the best choice as it yields a reasonable number of BSs per cluster. In other words, the MNO has to choose $r$ based on the size of its network since different network sizes need different values for $r$. As described in Table 2, first, we use the *elbow method* [54] to determine the number of clusters $r$, where $r = 100$ is optimal and the clustering process takes 343.76 seconds. However, when $r = 100$, we have many clusters that have many BSs and, this increases the communication delay among MEC servers belonging to the same collaboration space. To overcome this challenge, we increase the value of $r$ to 500 which still leads to many clusters that have many BSs. Subsequently, as shown in Fig. 4, we increase $r$ to 1000, at which point the average number of BSs per one collaboration space becomes 13 BSs. However, for values of $r$ larger than 1000, e.g., for $r = 2000$, the system has many small clusters (e.g., with only one BS). Among 1000 collaboration spaces, we randomly select one collaboration space, which has 12 BSs, and we associate each BS with 1 MEC server. Furthermore, we consider the initial number of users to be $K = 100$ at each BS, and we exponentially increase the number of users to $K = 3200$. In our setup, each user sends one task at each time slot. The path loss factor is set to 4 and the transmission power is set to $\rho_k = 27.0$ dBm [28], while the channel bandwidth is set to be in the range from $B_m = 25$ MHz to $B_m = 32$ MHz [53]. Furthermore, we consider the bandwidth between each pair of BSs to be randomly selected in the range from $\Gamma_m^n = 20$ MHz to $\Gamma_m^n = 25$ MHz, while the bandwidth between each BS and DC is selected in the range from $\Omega_m^{DC} = 50$ to $\Omega_m^{DC} = 120$ Mbps. The cache storage of each MEC server $m$ is in the range from 100 to 500 TB, while computation resources are in the range from 2 GHz to 2.5 GHz[55].

For a task $T_k$ of a given user $k$, we generate synthetic data. The size of the data $s(d_k)$ is randomly generated within a range of 2 to 7 GB, while the task computation deadline $\tilde{\tau}_k$ is randomly generated within a range of $\tilde{\tau}_k = 0.02$ second to $\tilde{\tau}_k = 12$ seconds. The workload $z_k$ of each user device $k$ is randomly generated and uniformly distributed in the range from $z_k = 452.5$ cycles/bit to $z_k = 737.5$ cycles/bit [55]. For each user device, the computation resource is in range from 0.5 GHz to 1.0 GHz [56].

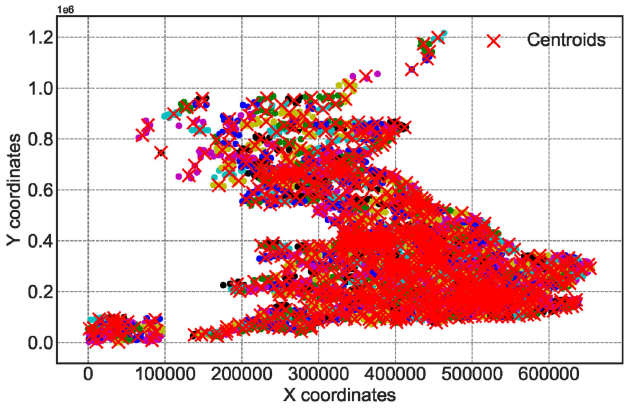


Fig. 4. Collaboration space formation ($r = 1000$).

The number of requests for contents ranges from $\lambda_m^{d_k} = 578$ to $\lambda_m^{d_k} = 3200$. The demand and popularity of the content follow Zipf distributions described in [57], [58].

### 5.2 Performance Metrics

#### 5.2.1 Throughput

For effective resource utilization, we evaluate the network and computation throughputs of the proposed algorithms. We consider the network throughput as a measurement of how many data units the network can handle within a given period of time [59], [60]. Meanwhile, the computation throughput is defined as a measurement of how many task units the MEC server can compute within a given period of time. Here, the network throughput is measured in terms of Mbps, while the computation throughput is measured in terms of million instructions per second (MIPS).

#### 5.2.2 Delay

In a collaboration space, each task $T_k$ offloaded by the user device ends its journey at the server which has resources that can fulfill user demand. Then, the MEC server computes, caches, and returns the output of the computation to the user. Therefore, we consider the total delay as the time period between offloading task $T_k$ and receiving the corresponding computation output. Thus, the total delay does not allow to visualize offloading delay and computation delay separately, we use transmission delay and computation/executing delay described in Section 4 as delay metrics.

#### 5.2.3 Cache Hit Ratio and Bandwidth-Saving

We also evaluate the number of cache hits and misses. A cache hit, denoted $h_m^{d_k} \in \{0, 1\}$, occurs when the requested content $d_k$ is retrieved from the cache storage available in a collaboration space at any BS $m$. Cache hit contributes to bandwidth saving defined in (27) as it reduces the data exchange between the collaboration space and the DC. On the other hand, a cache miss occurs when the requested content $d_k$ is not available in any cache storage in the collaboration space. The probability of a cache hit for content $d_k$ is expressed as follows:

$$P_{d_k} = \frac{\sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{M}} h_m^{d_k}}{\sum_{k \in \mathcal{K}} \sum_{m \in \mathcal{M}} (h_m^{d_k} + (1 - h_m^{d_k}))}, \tag{47}$$

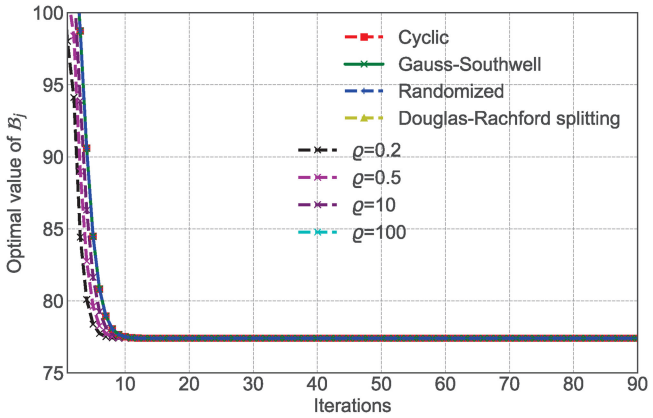


Fig. 5. Optimal value of $\mathcal{B}_j$ (35) with different coordinate selection rules (without rounding).

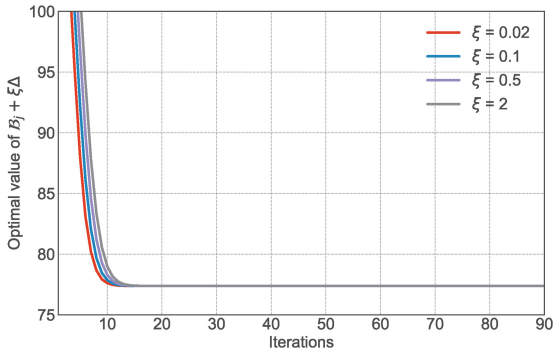


Fig. 6. Optimal value of $\mathcal{B}_j + \xi\Delta$ with different coordinate selection rules (after rounding).

where $\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}} h_m^{d_k}$ is the total number of cache hits, and $\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}(h_m^{d_k} + (1 - h_m^{d_k}))$ is the total number of cache hits plus the total number of cache misses.

### 5.3 Simulation Results

Fig. 5 combines both delay viewed as cost and bandwidth saving in one optimization problem in (35). We solve the proximal upper-bound problem through the use of distributed optimization control algorithm for 4C (Algorithm 3). Furthermore, we compare the solution of our distributed optimization control algorithm with the solution computed via Douglas-Rachford Splitting (D-R-S) [61] without applying a rounding technique. Thus, our formulated problem in (35) is decomposable. The Douglas-Rachford splitting method is used to decompose our problem into small subproblems, and address each subproblem separately. For any given two functions $f$ and $g$, the D-R-S method minimizes $f(x) + g(x)$ by using the following process: At the first iteration $t = 0$, it starts at an initial feasible $y^{(0)}$ and keeps updating $x$ and $y$ such that $x^{(t)} = \text{prox}_f(y^{(t-1)})$ and $y^{(t)} = y^{(t-1)} + \text{prox}_g(2x^{(t)} - y^{(t-1)}) - x^{(t-1)}$, where $\text{prox}_f$ and $\text{prox}_g$ are proximal functions of $f$ and $g$ [61], respectively.

Fig. 5 shows the convergence of our optimization problem. In this figure, we use the Douglas-Rachford Splitting method [61] and our distributed control algorithm (Algorithm 3) for solving (35). In our distributed control algorithm, for choosing indexes in (35), we use three coordinate selection rules: Cyclic, Gauss-Southwell, and Randomized [44]. Furthermore, for the quadratic term in (35), we adjust the positive penalty parameter $\varrho_j$ within the range 0.2 to 100. From this figure, we can see that the performance of our distributed control algorithm and Douglas-Rachford splitting method is almost the same. Therefore, the proximal upper-bound problem in (35) converges to both a coordinate-wise minimum and a stationary point, which is considered as a solution of (35). In other words, we consider this minimum point as an optimal value and equilibrium/stability point of $\mathcal{B}_j$ (35).

In Fig. 6, we apply the rounding technique to the results of Fig. 5 and solve $\mathcal{B}_j + \xi\Delta$, where we consider the positive rounding threshold to be $\theta = 7$ and the weight parameter $\xi$ of $\Delta$ is within the range 0.02 to 2.0. The simulation results in Fig. 6 ensure that the relaxed $\boldsymbol{x}_j$, $\boldsymbol{y}_j$, and $\boldsymbol{w}_j$ to be vectors of binary variables, and the rounding technique does not violate the computational and caching resource constraints while solving $\mathcal{B}_j + \xi\Delta$. Furthermore, the difference between Figs. 5 and 6 resides in the sizes of the problems ($\mathcal{B}_j$ and $\mathcal{B}_j + \xi\Delta$) and the step sizes needed for reaching the minimum point. However, in both Figs. 5 and 6, both problems $\mathcal{B}_j$ and $\mathcal{B}_j + \xi\Delta$ converge to the same stability point. In other words, with and without applying rounding technique, (35) converges to a minimum point that guarantees $\beta = 1$ ( no violations of communication, computational, and caching resources constraints).

In terms of network throughput, Fig. 7 shows that the throughput increases up to 35 Mbps. In this figure, the coordinate selection rules (Cyclic, Gauss-Southwell, Randomized) in our distributed optimization control algorithm and the Douglas-Rachford splitting method have almost the same performance.

Fig. 8 shows the cumulative distribution function (CDF) of the computational throughput. The simulation results show that the Cyclic selection rule in our distributed optimization control algorithm, as well as the Douglas-Rachford splitting (D-R-S) method, require high computational resources, as the computational throughput for each MEC server can reach $2.55 \times 10^8$ MIPS. On the other hand, the Gauss-Southwell and Randomized selection rules use less computational resources, as the computational throughput for each MEC server can reach $1.48 \times 10^8$ MIPS. The advantage of the Gauss-Southwell selection rule compared to other coordinate selection rules lies in choosing the index. In the Gauss-Southwell selection rule, instead of choosing the index randomly or cyclically, at each iteration, an index that maximizes the utilization of the computational resource is chosen.

We next examine the total delay between offloading task $T_k$ and receiving the corresponding computation output. Fig. 9 shows the transmission delay, where the solid blue lines represent the median and the dashed black lines represent the arithmetic mean. In this figure, Cyc stands for Cyclic, G-S stands for Gauss-Southwell, Ran stands for Randomized, and D-R-S stands for Douglas-Rachford splitting. The results in this figure show that the mean of the transmission delay varies from 0.0078 (G-S) to 0.092 (Cyc) seconds. In addition, Fig. 10 shows computation delay, where the mean of the computation delay varies from 0.008 (G-S) to 0.142 (D-R-S) seconds. The total delay, i.e., the sum of computation and transmission delays, fulfills the task computation deadline described in the simulation setup. However, Cyclic and Douglas-Rachford splitting yield higher delay than others due to index selection (for Cyclic) and splitting (for Douglas-Rachford splitting), which require more time and computation resources. Furthermore, Douglas-Rachford splitting has a higher delay than BSUM coordinate selection rules.

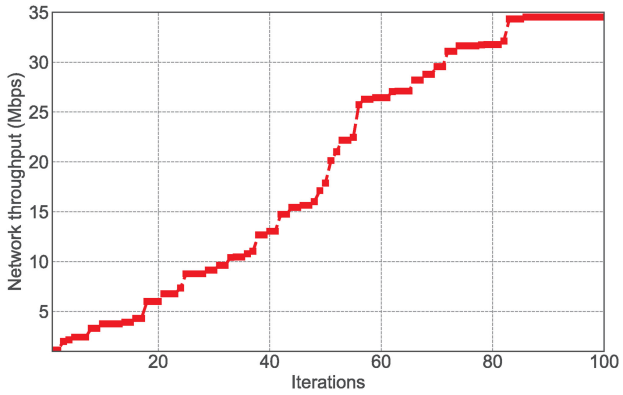


Fig. 7. Network throughput within a collaboration space.

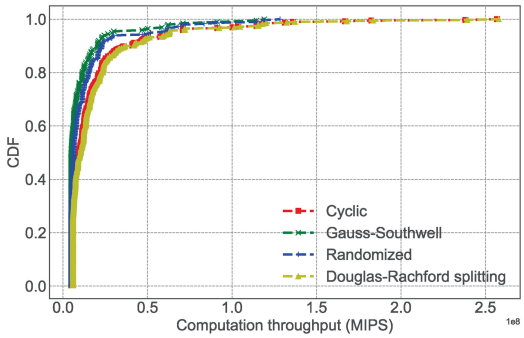


Fig. 8. CDF of computation throughput.

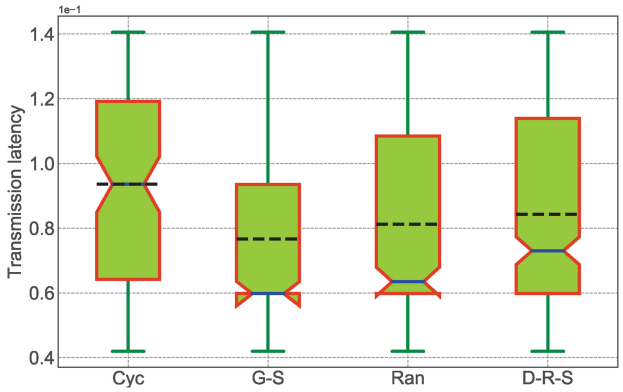


Fig. 9. Transmission delay.

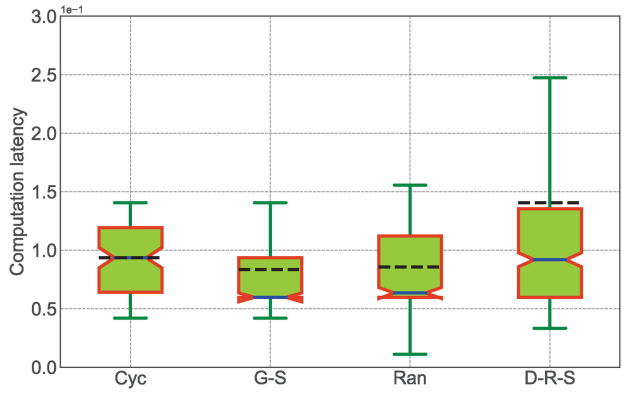


Fig. 10. Computation delay.

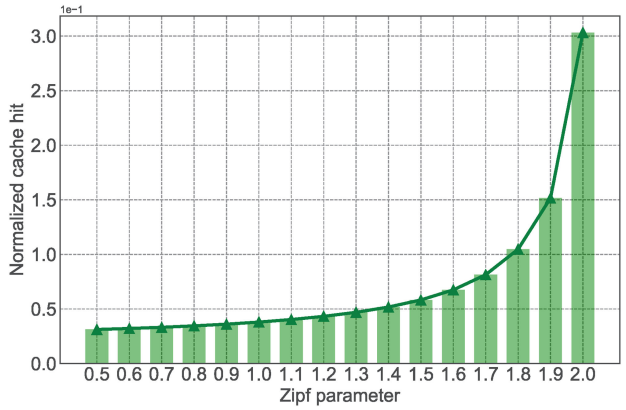


Fig. 11. Normalized cache hits in collaboration space.

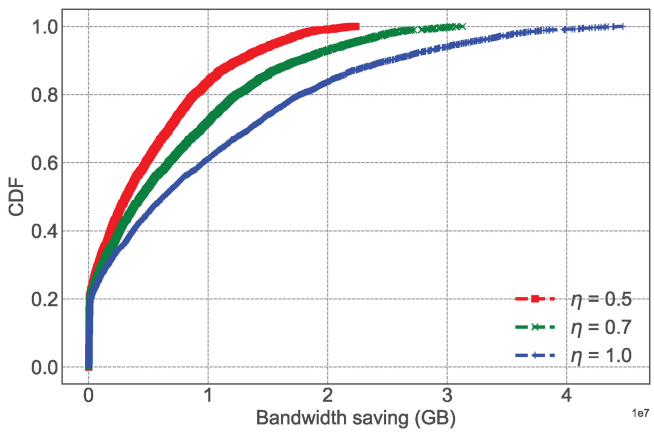


Fig. 12. Bandwidth saving due to caching.

Furthermore, in Figs. 5, 6, 8, 9, and 10, the only difference between D-R-S and other techniques is the computational resource utilization and computation delay. Therefore, since computational resources are limited and each offloaded task $T_k$ has computation deadline, these results clearly demonstrate that the proposed distributed optimization control algorithm (BSUM-based) for 4C is more suitable than D-R-S.

Fig. 11 shows the normalized cache hits, where cache hit ratio $P_{d_k}$ is computed from (47). From Fig. 11, we can see that the cache hit ratio increases with the Zipf exponent parameter $a$. When $a = 2.0$, due to the increase in the number of demands for contents, many contents become popular, which results in a high cache hit ratio of 0.03 percent of the total demands $\lambda_m^{d_k}$ from users. In the case of cache misses in collaboration space, the demands for contents need to be forwarded to the DC. Therefore, cache hits contribute to reducing the number of demands $\lambda_m^{d_k}$ for contents that need to be forwarded to the DC. Furthermore, using the number of demands $\lambda_m^{d_k}$ and the size of cached contents $d_k$ in collaboration space, we compute bandwidth-saving through the use of (27).

Fig. 12 shows the simulation results for bandwidth-saving in terms of Gigabytes (GB). In this figure, from the beginning, bandwidth-saving is nearly zero, and thus MEC server has to cache the contents first. In other words, MEC caching is based on content prefetching. Therefore, due to the increase in the number of cached contents and demands, the maximum bandwidth-saving of $4.74 \times 10^7$ GB is observed when $a = 2.0$ and $\eta = 1$. Furthermore, the increase in the demands is accompanied by an increase in network throughput, cache storage, computational resource utilization, and delay.

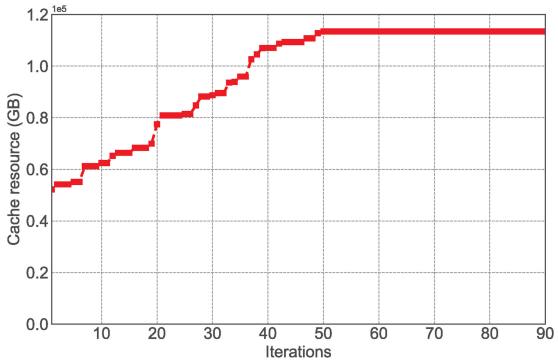


Fig. 13. Utilization of MEC cache storage units.

Fig. 13 shows the total cache storage utilization in the collaboration space of 12 MEC servers, where the cache storage utilization depends on the sizes of offloaded data and cache capacity constraints. In Fig. 13, we can see that the cache resources utilization increases with the number of demands until it reaches to $1.15 \times 10^5$ GB (when $a = 1.0$). The increase of cache storage utilization results in the increase of cache hits in Fig. 11 and bandwidth saving in Fig. 12.

In this work, our approach focuses on intra-cooperation between MEC servers that belong to one collaboration space. One interesting future work is to extend our framework to account for inter-cooperation between MEC servers that belong to different collaboration spaces.

## 6 Conclusion

In this paper, we have proposed a joint communication, Communication, Computation, Caching, and Control (4C) framework for big data MEC. In this framework, MEC servers collaborate to satisfy users' demand. We have formulated the problem as a joint optimization problem that aims to minimize a linear combination of bandwidth consumed and network latency. Therefore, for solving the formulated optimization problem, we have proposed a distributed optimization control algorithm for 4C, which is a modified version of the BSUM method. We have compared the results from the distributed optimization control algorithm with the results computed via the Douglas-Rachford splitting method. Simulation results from both methods have shown that our approach can be efficiently implemented.

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2017R1A2A2A05000995).

## References

[1] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 22–25, 2015, pp. 167–176.

[2] E. Dave, "The internet of things: How the next evolution of the internet is changing everything.," *CISCO white paper* 1, no. 2011, Apr. 2011, pp. 1–11. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[3] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sep. 16, 2016.

[4] S. Ranadheera, S. Maghsudi, and E. Hossain, "Computation offloading and activation of mobile edge computing servers: A minority game," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 688–691, Oct. 2018.

[5] A. Ferdowsi, U. Challita, and W. Saad, "Deep learning for reliable mobile edge analytics in intelligent transportation systems," *CoRR*, Dec. 2017. [Online]. Available: https://arxiv.org/pdf/1712.04135.pdf.

[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 5 Sep. 2015.

[7] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al., "Mobile-edge computing introductory technical white paper," White Paper, Mobile-edge Computing (MEC) Industry Initiative, Sep. 2014.

[8] O. Semiari, W. Saad, S. Valentin, M. Bennis, and H. V. Poor, "Context-aware small cell networks: How social metrics improve wireless resource allocation," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 5927–5940, Jul. 13, 2015.

[9] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 14, 2017.

[10] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, "Collaborative cache allocation and computation offloading in mobile edge computing," in *Proc. 19th IEEE Asia-Pacific Netw. Operations Manag. Symp.*, Sep. 27–29, 2017, pp. 366–369.

[11] K. Dutta and M. Jayapal, "Big data analytics for real time systems," in *Proc. Big Data Analytics Seminar*, Nov. 11, 2015, pp. 1–13.

[12] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Generation Comput. Syst.*, vol. 70, May 2017, pp. 59–63.

[13] G. Cleuziou, "An extended version of the K-means method for overlapping clustering," in *Proc. 19th IEEE Int. Conf. Pattern Recognit.*, Dec. 08–11, 2008, pp. 1–4.

[14] Z. Han, M. Hong, and D. Wang, *Signal Processing and Networking for Big Data Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[15] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *J. Commun. Netw.*, vol. 17, no. 6, pp. 549–557, Dec. 2015.

[16] E. Bastug, K. Hamidouche, W. Saad, and M. Debbah, "Centrality-based caching for mobile wireless networks," in *Proc. 1st KuVS Workshop Anticipatory Netw.*, Sep. 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01094823/.

[17] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *CoRR*, vol. abs/1710.02913, Oct. 2017. [Online]. Available: https://arxiv.org/pdf/1710.02913.pdf.

[18] W. Fan, Y. Liu, B. Tang, F. Wu, and H. Zhang, "Terminalbooster: Collaborative computation offloading and data caching via smart basestations," *IEEE Wireless Commun. Lett.*, vol. 5, no. 6, pp. 612–615, 02 Sep. 2016.

[19] G. Lee, W. Saad, and M. Bennis, "An online secretary framework for fog network formation with minimal latency," *CoRR*, vol. abs/1702.05569, Apr. 2017. [Online]. Available: https://arxiv.org/pdf/1702.05569.pdf.

[20] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. 13th IEEE Annu. Conf. Wireless On-demand Netw. Syst. Serv.*, 21–24 Feb. 21–24, 2017, pp. 165–172.

[21] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, "Mobility-aware caching and computation offloading in 5G ultra-dense cellular networks," *Sensors*, vol. 16, no. 7, Jun. 25, 2016, Art. no. 974.

[22] F. Zhang, C. Xu, Y. Zhang, K. Ramakrishnan, S. Mukherjee, R. Yates, and T. Nguyen, "Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture," in *Proc. IEEE 16th Int. Symp. World Wireless Mobile Multimedia Netw.*, Jun. 14–17, 2015, pp. 1–9.

[23] X. Vasilakos, V. A. Siris, and G. C. Polyzos, "Addressing niche demand based on joint mobility prediction and content popularity caching," *Comput. Netw.*, vol. 110, pp. 306–323, 2016.

[24] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, 03 Aug. 2016.

[25] H. Hsu and K.-C. Chen, "A resource allocation perspective on caching to achieve low latency," *IEEE Commun. Lett.*, vol. 20, no. 1, pp. 145–148, 09 Nov. 2015.

[26] Z. Tan, X. Li, F. R. Yu, L. Chen, H. Ji, and V. C. Leung, "Joint access selection and resource allocation in cache-enabled HCNs with D2D communications," in *Proc. IEEE Wireless Commun. Netw. Conf.*Mar. 19–22, 2017, pp. 1–6.

[27] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, Mar. 9, 2017.

[28] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11 339–11 351, Aug. 09, 2017.

[29] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, pp. 4924–4938, May 16, 2017.

[30] R. Huo, F. R. Yu, T. Huang, R. Xie, J. Liu, V. C. Leung, and Y. Liu, "Software defined networking, caching, and computing for green wireless networks," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 185–193, Nov. 15, 2016.

[31] J. Chakareski, "VR/AR immersive communication: Caching, edge computing, and transmission trade-offs," in *Proc. ACM Workshop Virtual Reality Augmented Reality Netw.*, Aug. 21–25, 2017, pp. 36–41.

[32] Y. Cui, W. He, C. Ni, C. Guo, and Z. Liu, "Energy-efficient resource allocation for cache-assisted mobile edge computing," *CoRR*, vol. abs/1708.04813, Aug. 16, 2017. [Online]. Available: https://arxiv.org/pdf/1708.04813.pdf.

[33] S. Baadel, F. Thabtah, and J. Lu, "Overlapping clustering: A review," in *Proc. IEEE Comput. Conf.*, Jul. 3–15, 2016, pp. 233–237.

[34] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige, and R. Buyya, "The anatomy of big data computing," *Softw.: Practice Experience*, vol. 46, no. 1, pp. 79–105, Jan. 2016.

[35] T. Nguyen and M. Vojnovic, "Weighted proportional allocation," in *Proc. ACM Joint Int. Conf. Meas. Model. Comput. Syst.*, Jun. 07–11, 2011, pp. 173–184.

[36] S. Mosleh, L. Liu, and J. Zhang, "Proportional-fair resource allocation for coordinated multi-point transmission in LTE-advanced," *IEEE Trans. Wireless Commun.*, vol. 15, no. 8, pp. 5355–5367, Apr. 21, 2016.

[37] L. Lei, D. Yuan, C. K. Ho, and S. Sun, "Joint optimization of power and channel allocation with non-orthogonal multiple access for 5G cellular systems," in *Proc. IEEE Global Commun. Conf.*, Dec. 6–10, 2015, pp. 1–6.

[38] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 25 Aug. 2017.

[39] C. B. Networks, "Backhauling x2," [Online]. Available: http://cbnl.com/resources/backhauling-x2, Accessed on: Feb. 3, 2018. [Online]. Available: https://cbnl.com/resources/backhauling-x2.

[40] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Proc. IEEE Eur. Conf. Netw. Commun.*, Jun. 12–15, 2017, pp. 1–6.

[41] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.

[42] A. Ndikumana, S. Ullah, and C. S. Hong, "Scalable aggregation-based packet forwarding in content centric networking," in *Proc. 18th IEEE Asia-Pacific Netw. Operations Manag. Symp.*, Nov. 10, 2016, pp. 1–4.

[43] A. Ndikumana, N. H. Tran, T. M. Ho, D. Niyato, Z. Han, and C. S. Hong, "Joint incentive mechanism for paid content caching and price based cache replacement policy in named data networking," *IEEE Access*, vol. 6, pp. 33 702–33 717, 2018.

[44] M. Hong, M. Razaviyayn, Z.-Q. Luo, and J.-S. Pang, "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 57–77, 25 Dec. 2015.

[45] M. Hong, T.-H. Chang, X. Wang, M. Razaviyayn, S. Ma, and Z.-Q. Luo, "A block successive upper bound minimization method of multipliers for linearly constrained convex optimization," *CoRR*, Jun. 2014. [Online]. Available: https://arxiv.org/pdf/1401.7079.pdf.

[46] A. Ndikumana, N. H. Tran, and C. S. Hong, "Deep learning based caching for self-driving car in multi-access edge computing," *CoRR*, vol. abs/1810.01548, Oct. 2018. [Online]. Available: https://arxiv.org/pdf/1810.01548.pdf.

[47] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo, "Iteration complexity analysis of block coordinate descent methods," *Math. Program.*, vol. 163, no. 1–2, pp. 85–114, May 2017.

[48] U. Feige, M. Feldman, and I. Talgam-Cohen , "Oblivious rounding and the integrality gap," in *Proc. Leibniz Int. Proc. Inform.*, Dec. 13–16, 2016, 1–23.

[49] N. Zhang, Y.-F. Liu, H. Farmanbar, T.-H. Chang, M. Hong, and Z.-Q. Luo, "Network slicing for service-oriented networks under resource constraints," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2512–2521, Oct. 05, 2017.

[50] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Jul. 25, 2017.

[51] M. Farivar, X. Zho, and L. Che, "Local voltage control in distribution systems: An incremental control algorithm," in *Proc. Int. Conf. Smart Grid Commun.*, Nov. 2–5, 2015, pp. 732–737.

[52] G. Van Rossum, et al., "Python programming language," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 17–22, 2007, vol. 41, Art. no. 36.

[53] O. Boswarva, "Sitefinder mobile phone base station database," Feb. 2017. [Online]. Available: https://datashare.is.ed.ac.uk/handle/10283/2626.

[54] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in K-means clustering," *Int. J.*, vol. 1, no. 6, pp. 90–95, 2013.

[55] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Jun. 23, 2017.

[56] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 26, 2015.

[57] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary Phys.*, vol. 46, no. 5, pp. 323–351, Feb. 20, 2007.

[58] A. Ndikumana, K. Thar, T. M. Ho, N. H. Tran, P. L. Vo, D. Niyato, and C. S. Hong, "In-network caching for paid contents in content centric networking," in *Proc. IEEE Global Commun. Conf.*, Dec. 4–8, 2017, pp. 1–6.

[59] A. Ndikumana, S. Ullah, K. Thar, N. H. Tran, B. J. Park, and C. S. Hong, "Novel cooperative and fully-distributed congestion control mechanism for content centric networking," *IEEE Access*, vol. 5, pp. 27 691–27 706, Nov. 29, 2017.

[60] A. Ndikumana, S. Ullah, R. Kamal, K. Thar, H. S. Kang, S. I. Moon, and C. S. Hong, "Network-assisted congestion control for information centric networking," in *Proc. 17th IEEE Asia-Pacific Netw. Operations Manag. Symp.*, Aug. 19–21, 2015, pp. 464–467.

[61] A. Themelis, L. Stella, and P. Patrinos, "Douglas-rachford splitting and ADMM for nonconvex optimization: New convergence results and accelerated versions," *CoRR*, vol. 1709.05747v2, Jan. 2018. [Online]. Available: https://arxiv.org/pdf/1709.05747.pdf.

**Anselme Ndikumana** received the BS degree in computer science from the National University of Rwanda, in 2007. He is currently working toward the PhD degree at the Department of Computer Science and Engineering, Kyung Hee University, South Korea. His professional experience includes being a chief information officer, a system analyst, and a database administrator at the Rwanda Utilities Regulatory Authority from 2008 to 2014. His research interest includes multi-access edge computing, deep learning, named data networking, and in-network caching.

**Nguyen H. Tran** (S'10-M'11) received the BS degree in electrical and computer engineering from the Hochiminh City University of Technology, in 2005, and the PhD degree in electrical and computer engineering from Kyung Hee University, in 2011. He was an assistant professor with the Department of Computer Science and Engineering, Kyung Hee University from 2012 to 2017. Since 2018, he has been with the School of Computer Science, The University of Sydney, where he is currently a senior lecturer. His research interest is applying analytic techniques of optimization, game theory, and machine learning to cutting-edge applications such as cloud and mobile-edge computing, datacenters, resource allocation for 5G networks, and Internet of Things. He received the best KHU thesis award in engineering in 2011 and several best paper awards, including IEEE ICC 2016, APNOMS 2016, and IEEE ICCS 2016. He received the Korea NRF Funding for Basic Science and Research from 2016 to 2023. He has been the editor of the *IEEE Transactions on Green Communications and Networking* since 2016. He is a senior member of the IEEE.

**Tai Manh Ho** received the BEng and MS degrees in computer engineering from the Hanoi University of Science and Technology, Vietnam, in 2006 and 2008, respectively, and the PhD degree in computer engineering from Kyung Hee University, South Korea, in 2018. Since 2018, he has been with the Institut National de la Recherche Scientifique (INRS), Universite du Quebec, Monteral, QC, Canada, where he is currently a postdoctoral fellow. His current research interests include radio resource management and enabling technologies for 5G wireless systems.

**Zhu Han** (S'01-M'04-SM'09-F'14) received the BS degree in electronic engineering from Tsinghua University, in 1997, and the MS and PhD degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was an R&D Engineer of JDSU, Germantown, Maryland. From 2003 to 2006, he was a research associate with the University of Maryland. From 2006 to 2008, he was an assistant professor at Boise State University, Idaho. Currently, he is a John and Rebecca Moores professor with the Electrical and Computer Engineering Department as well as in the Computer Science Department, University of Houston, Texas. He is also a chair professor at National Chiao Tung University, ROC. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the Journal on Advances in Signal Processing in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (best paper award in IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. He was an IEEE Communications Society Distinguished lecturer from 2015-2018. He has been a 1 percent highly cited researcher since 2017 according to Web of Science. He is a fellow of the IEEE.

**Walid Saad** (S'07-M'10-SM'15) received the PhD degree from the University of Oslo, Oslo, Norway, in 2010. He is currently an associate professor with the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, where he leads the Network Science, Wireless, and Security Laboratory, Wireless@VT Research Group. His current research interests include wireless networks, machine learning, game theory, cybersecurity, unmanned aerial vehicles, and cyber-physical systems. He was a recipient of the NSF CAREER Award in 2013, the AFOSR Summer Faculty fellowship in 2014, the Young Investigator Award from the Office of Naval Research in 2015, the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, the 2017 IEEE ComSoc Best Young professional in Academia Award, and six conference Best Paper Awards at WiOpt in 2009, ICIMP in 2010, IEEE WCNC in 2012, IEEE PIMRC in 2015, IEEE SmartGridComm in 2015, and EuCNC in 2017. From 2015 to 2017, he was named the Stephen O. Lane Junior Faculty fellow at the Virginia Polytechnic Institute and State University and, in 2017, he was named the College of Engineering Faculty fellow. He currently serves as an editor for the *IEEE Transactions on Wireless Communications*, the *IEEE Transactions on Communications*, the *IEEE Transactions on Mobile Computing*, and the *IEEE Transactions on Information Forensics and Security*. He is a fellow of the IEEE.

**Dusit Niyato** (M'09-SM'15-F'17) received the BEng degree from the King Mongkuts Institute of Technology Ladkrabang, in 1999, and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada, in 2008. He is currently a full professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include the areas of green communication, the Internet of Things, and sensor networks. He is a fellow of the IEEE.

**Choong Seon Hong** received the BS and MS degrees in electronic engineering from Kyung Hee University, Seoul, South Korea, in 1983 and 1985, respectively, and the PhD degree from Keio University, in 1997. In 1988, he joined KT, where he was involved in broadband networks, as a member of technical staff. In 1993, he joined Keio University, Japan. He was with the Telecommunications Network Laboratory, KT, as a senior member of technical staff and as the director of the Networking Research Team until 1999. Since 1999, he has been a professor in the Department of Computer Science and Engineering, Kyung Hee University. His research interests include future internet, ad hoc networks, network management, and network security. He has served as a general chair, TPC chair/member, or an organizing committee member for international conferences, such as NOMS, IM, APNOMS, E2EMON, CCNC, ADSN, ICPP, DIM, WISA, BcN, TINA, SAINT, and ICOIN. He is currently an associate editor of the *International Journal of Network Management,* and the *IEEE Journal of Communications and Networks*, and an associate technical editor of the *IEEE Communications Magazine*. He is a member of the ACM, the IEICE, the IPSJ, the KIISE, the KICS, the KIPS, and the OSIA. He is a senior member of the IEEE.